Name:

COMPSCI 111/111G

An Introduction to Practical Computing

COMPSCI 111/111G
Computer Science Department
The University of Auckland

COMPSCI 111/111G
An Introduction to Practical Computing

Reference Manual

June 10, 2016

Andrew Luxton-Reilly

Copyright © 2008

Copyright Notice

This coursebook may be used only for the University's educational purposes. It includes extracts of copyright works copied under copyright licences. You may not copy or distribute any part of this coursebook to any other person. Where this coursebook is provided to you in electronic format you may only print from it for your own use. You may not make a further copy for any other purpose. Failure to comply with the terms of this warning may expose you to legal action for copyright infringement and/or disciplinary action by the University.

Acknowledgements

Thanks to everyone who contributed to the development of this resource. Particular thanks go to Ann Cameron for extensive feedback and endless proofreading.

Feedback

We welcome any and all feedback on the coursebook. Please send all corrections, comments and other feedback on this coursebook to andrew@cs.auckland.ac.nz

Contents

Learning Outcomes

Bits, bytes and binary numbers

Students will be familiar with the binary representation of numbers and the prefixes commonly used with binary numbers.

Students should be able to:

- define the terms “bit” and “byte”.
- use the decimal SI units (kilo, mega, giga, and tera) appropriately.
- compare and contrast the binary SI units (kibi, mebi, gibi, tebi) with the corresponding decimal SI units.
- state the biggest decimal number that can be represented using a given number of binary digits.
- state the number of binary digits required to represent a given decimal number.

Standards

Students will have an appreciation of how numbers can be used to represent different kinds of information (such as text and images). Students will understand the importance of standard methods of encoding.

Students should be able to:

- describe how numbers can be used to encode text and images.
- compare and contrast open standards with proprietary standards, giving examples of each.

Hardware

Students will understand the purpose of the major components of a computer system, and will be able to identify those components visually.

Students should be able to:

- assign common hardware to one of the categories “Input”, “Output”, “Processing”, “Storage” and “Communication”.
- identify the major components of a computer system.
- explain the purpose of each of the hardware components found in a typical desktop computer.
- state Moore’s Law and discuss the implications for computing.
- compare and contrast primary memory with mass storage devices.
- describe the major factors influencing the performance of a computer, and explain the different ways that computer performance is measured.
- read an advertisement for a computer system and explain what it means.

Software

Students will be aware of some major software companies. Students will have an understanding of different software licences, and the purpose of common system and application software.

Students should be able to:

- distinguish between software and hardware.
- identify some major software companies and the products that they create.
- distinguish between application software and system software.
- describe the different categories of software licences.

Operating systems

Students will be aware of the major operating systems available today, will be able to explain the purpose of an operating system and will be able to use a standard operating system comfortably.

Students should be able to:

- identify common file extensions and the applications they correspond to.
- give examples of different operating systems.
- discuss the purpose of an operating system.

User interfaces

Students will understand the purpose of a user interface and will be aware of the differences between command line interfaces and graphical user interfaces.

Students should be able to:

- explain the meaning of the acronyms GUI and CLI.
- compare and contrast a GUI with a CLI.
- use the correct terminology to identify parts of a graphical user interface.

Internet

Students will have a basic understanding of the development of the Internet, and will be familiar with simple networking terminology.

Students should be able to:

- explain the meaning and purpose of TCP/IP.
- put a series of Internet related events into chronological order.
- describe some of the design features of the Internet and explain why it was designed the way it was.
- state the purpose of the DNS and describe how it works.
- describe the purpose of networking components required for a home network — network card, modem, router.
- describe,in simple terms, how information is transferred through the Internet.

WWW

Students will be familiar with the development of the WWW, its relationship to the Internet and how to effectively use the WWW.

Students should be able to:

- put a series of events related to the WWW into chronological order.
- describe the difference between the WWW and the Internet.
- describe the underlying process that occurs when a user looks at a web page.
- describe the way that web page access is logged.
- discuss how search engines rank pages
- discuss the implications arising from our use of search engines to access the WWW.
- describe some of the copyright issues that relate to the use of search engines.

- define the following terms/acronyms: www, http, hypertext, hypermedia, url.
- use a search engine to find information on the WWW.

Electronic communication — Email, IM, Forums

Students will be familiar with different tools used to communicate online.

Students should be able to:

- explain the purpose of the common header fields — To, From, Reply-to, CC, BCC, Subject.
- compare and contrast IMAP and POP3.
- describe how an email message is transferred from the sender to the receiver.
- discuss issues around the privacy of email and the use of email in the workplace.
- state some of the benefits and dangers present in electronic communication.
- compare and contrast different forms of communication — Email, IM, Forums.
- give examples of good and bad netiquette.
- explain what spam is and why it is undesirable.
- describe some of the tools that are typically included with an email system — address books, filters.
- define (with examples) common terminology used with electronic communication systems — threads, moderators, flames, quotes, emoticons, acronymns.
- use webmail to send and receive email messages.
- create and use address book entries.
- read and post messages to an electronic forum.

Online community tools — Blogs, Wikis

Students will have an appreciation of the tools that are commonly used by online communities.

Students should be able to:

- describe what a blog is.
- discuss the social implications of blogs.
- compare and contrast the different tools used to publish information online — Forums, Blogs, Wikis.
- describe what a wiki is.
- discuss the accuracy of information on a wiki.

- explain how a community can effectively maintain a wiki.
- describe the common tools that are used within a wiki.
- create a blog.
- add a new posting to a blog.
- use wiki markup to create a wiki page.
- contribute to an existing wiki.

Word processing

Students will be familiar with the idea of encoding text, and the importance of standards. Students will gain experience using a word processor to format documents.

Students should be able to:

- describe the meaning of the acronym “ASCII” and explain why ASCII is important.
- use ASCII to encode or decode text.
- explain the difference between a text editor and a word processor.
- distinguish between examples of surface formatting and examples of structural formatting.
- describe the advantages of structural formatting over surface formatting.
- use common formatting commands to format a document.
- create, modify and apply user-defined styles within a document.
- use EndNote to create citations.

Digital images

Students will have an appreciation of different methods of encoding digital images.

Students should be able to:

- describe how a bitmap is used to represent an image.
- describe how vector graphics are used to represent an image.
- compare and contrast bitmaps and vector graphics.
- calculate the size (in bytes) of a given bitmap image.
- compare and contrast jpeg and gif compression methods.
- use common drawing tools to create a diagram within a word processing document.

HTML5

Students will understand how web pages are created using a recent standard (HTML5), and will be able to create their own web pages using this standard.

Students should be able to:

- discuss the importance of using published standards.
- state what a Document Type Definition is used for.
- state the meaning of the acronym HTML.
- use HTML tags to create a web page that adheres to the HTML5 standard.
- validate web pages using an online validation tool.

CSS

Students will understand the advantages of using Cascading Style Sheets, and will be able to create a web page that uses both HTML5 and CSS.

Students should be able to:

- state the meaning of the acronym CSS.
- compare and contrast the different locations that styles can be defined.
- distinguish between and .
- distinguish between class and id selectors.
- write a style sheet that will produce a specified appearance (given a table of attributes).
- use style sheets to create a standard appearance for a web site that includes at least three web pages.

Web page design

Student will gain an appreciation of simple aspects of web page design.

Students should be able to:

- describe design that will aid navigation.
- discuss design decisions that arise with hypertext links.
- discuss the use of fonts, colour and backgrounds in web pages.

PowerPoint

Students will understand the functions and limitations of PowerPoint software, and will be able to use PowerPoint to create a presentation.

Students should be able to:

- state some of the criticisms of PowerPoint.
- identify aspects of good and bad presentation design.
- describe good use of structure and appearance (colour, backgrounds, font).
- use design templates, master slides, and animation to create a short presentation.

Spreadsheet

Students will understand how to create spreadsheets using cell references and simple functions.

Students should be able to:

- distinguish between absolute references and relative references.
- write formulae that calculate results based on the contents of other cells.
- write formulae that use common mathematical functions.
- evaluate boolean expressions.
- use IF, VLOOKUP and HLOOKUP functions.

Database

Students will understand what a relational database is, how to create and how to use a relational database.

Students should be able to:

- explain the difference between a database and a database management system.
- identify a field, record and table in a relational database.
- define and identify primary and foreign keys.
- use a relationship diagram to identify the relationships between different fields.
- create a simple relational database.
- compare and contrast QBE with SQL
- use QBE to create simple queries and run them on an existing database.
- write simple SQL queries that use SELECT, FROM, WHERE, ORDER BY and GROUP BY

Programming/Python

Students will understand simple programming concepts and be able to write very simple programs.

Students should be able to:

- compare and contrast interpreters with compilers.
- identify and use comments.
- distinguish between different types of information (strings, integers and floating point numbers).
- create expressions using standard mathematical operators.
- use the print statement to generate output.
- identify and use variables to store and recall values.
- read and store input from the user.
- write simple programs that read input, perform a calculation and produce output.
- evaluate boolean expressions that include comparison operators and logical operators.
- use an if statement.
- use a while loop to perform repetitive tasks.
- read a simple program and determine the output that would be produced.
- write simple programs that contain a while loop and/or if statements.

LaTeX

Students will be able to use LaTeX to typeset a document, including the typesetting of mathematical formulae.

Students should be able to:

- compare and contrast LaTeX with MS-Word.
- distinguish between the preamble and the document body.
- identify and use comments.
- use environments and simple commands
- distinguish between the three commonly used math modes.
- use math commands to typeset a complex mathematical formula.
- typeset a document that includes titles, columns, sections, footnotes, images and mathematical formulae.

History

Students will have a basic understanding of the history of the personal computer, the major companies and the people involved in that history.

Students should be able to:

- put a list of historically important events into chronological order.
- match the names of people with their accomplishments.
- explain why IBM failed to dominate the personal computer industry.
- explain how VISICALC and Apple are related.
- explain the relationship between Microsoft, MS-DOS, IBM and PC clones.

Social issues

Students will have an appreciation of some of the complex social issues that result from the Internet.

Students should be able to:

- explain how anonymous the Internet is.
- discuss advantages and disadvantages of anonymity.
- discuss issues of cultural dominance.
- describe the complexity of Internet censorship.
- discuss methods of protecting children from undesirable content on the Internet.
- compare and contrast different forms of malware — viruses, worms, trojans and logic bombs.

Acronyms

Students will be able to state the meaning of the following acronyms:

- CPU
- HDD, FDD
- RAM, ROM
- AGP
- ASCII
- GUI, CLI
- WYSIWYG
- OS

- TCP/IP, HTTP, FTP, SMTP, IMAP, POP3
- ISP
- DNS, WWW, URL
- JPEG, GIF, PNG, SVG
- HTML/XHTML, CSS
- DBMS, SQL, QBE

CHAPTER 1

Digital information

1.1 Analogue vs. Digital Systems

1.1.1 Analogue Systems

In an analogue system, information changes in a continuous manner. That is, values can change gradually in infinitely small steps. Examples for analogue systems are mechanic scales, tape measures, vinyl records and the brake and accelerator pedals of a car.

Figure 1.1: Both the tape measure and the dial are analogue devices

1.1.2 Digits

Digits are the building blocks of numbers. They are symbols such as “0”, “1”, “2” etc. which can be combined to express a quantity. If we combine the digits “1”, “0” and “3”

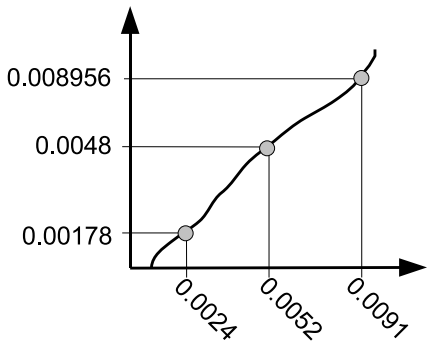


Figure 1.2: An analogue signal is continuous, i.e. it changes in infinitely small steps.

we get the number “one hundred and three”. The more digits we combine, the higher are the numbers that we can represent with them.

1.1.3 Digital Systems

In a digital system, information can only change within a fixed number of predefined, discrete steps. That is, only a certain number of values are actually possible. There are no in-between values as in the analogue systems, which have a continuous value range as described before. As a result, we can represent digital information precisely by using digits, hence the term “digital”. Examples for digital systems are computers, CD players, electronic scales and calculators.

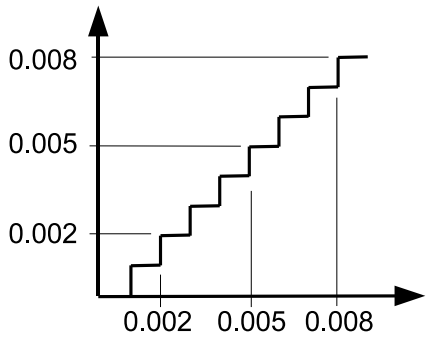


Figure 1.3: A digital system that has only 3 digits cannot represent numbers more precisely than .001.

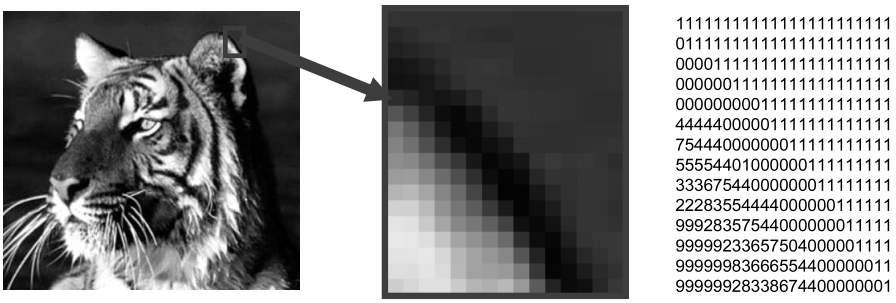
How many values a digital system can process depends on the maximum number of digits it can use. For example, most modern calculators have space to display around 10 different digits on the screen. The biggest number that could be displayed using these 10 digits is 9 999 999 999. The smallest number that could be represented using 10 digits is .000 000 000 1.

1.2 Encoding Information Digitally

Any information can be encoded using numbers. We just have to decide what encoding system to use. If we have analogue information and want to store it digitally, then we need to do something called *sampling*: sampling means that we divide the analogue information with its infinitely small changes into discrete blocks (*samples*) that can each be described by one value of a predefined set of values. In the following, let us consider how this works for images and sounds.

1.2.1 Encoding Images

In the real world, images consist of colours and shapes which change in infinitely small steps. That is, they are a type of analogue information and therefore change continuously. Devices that store images digitally, such as digital cameras, cannot cope with all these infinitely many colors and shapes. So they reduce an image to a finite number of building blocks called *pixels* (short for *pic*ture *el*ement). A pixel is a rectangular area that has a single colour. Each colour is encoded as a number using a certain number of digits. The reduction from the real image to the set of pixels is an example for sampling. As we can see in the figure below, the pixels are arranged in a matrix so that they look like the original image (although they do not contain all the infinitely small steps of the original anymore).

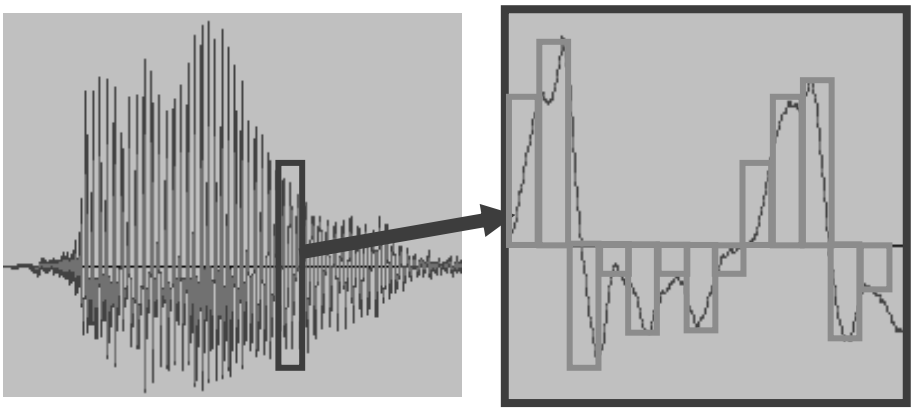


When magnifying the digital image on the left, we can see its pixels (middle), which are represented as a matrix of numbers (right).

1.2.2 Encoding Sounds

In the real world, sounds such as noises or music consist of vibrations in the air that make our eardrums vibrate as well. Since those vibrations are analogue information, they change continuously and in infinitely small steps. The vibrations can be described as a *waveform* over time, that is, as a description of how much our eardrum is stretched at each point in time. Such a waveform is shown on the left side of the figure below: time flows from left to right, and the zigzag deviations up and down signify the vibrations in the air. In order to make it possible to store sound on a digital device, such as an MP3 player, we need to break the waveform down into discrete steps that can each be described digitally with a number – another example of sampling. On the left side of the figure we see the magnified waveform and how the waveform is broken down into discrete steps by sampling. Each of the small boxes has the same width, i.e. we break

down the time into equal units (usually around $1/44000$ of a second), and the height of each approximates the degree to which our eardrum is stretched at that time. The boxes are simply called samples and are represented as a sequence of numbers, each number describing the height of one box.



Sound is recorded as a sequence of samples, each of which can be represented by a number.

When a sound is played by a digital device, the device goes through the samples in order and generates an electrical voltage with a strength proportional to the respective number. The electrical voltage is sent to a loudspeaker, and the membrane of the loudspeaker starts vibrating just as expressed by the samples. The membrane makes the air vibrate, which in turn makes our eardrums vibrate. Modern digital music players use clever tricks to store music with as little numbers as possible. The main idea behind music compression formats such as MP3 is that the human ear cannot hear all the samples anyway, so they do not store them all.

1.3 Binary Numbers

The way we usually represent our numbers is called the decimal representation. “Decimal” comes from the Latin word for “ten” and means that we use ten digits: “0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8” and “9”. The reason why we use ten digits is that we have ten fingers, which can be very convenient for counting. In fact, the word “digit” is also Latin and means “finger”. Computers store numbers using a different number representation, the binary representation. In the following we will learn how the binary representation of numbers works.

1.3.1 Storing Decimal Numbers in a Machine

Before learning how to use binary numbers, let us first consider how we would store decimal numbers in a machine such as a computer. For each of the decimal digits, we would need some sort of a dial that can be set to ten different states. Of course, in a computer we would not have mechanical dials but some sort of electronic storage units.

If we want to represent, say, decimal number with three digits (from 0 to 999) then

we need three such dials. The leftmost dial would store how many 100's there are in our number, the middle dial would store how many 10's there are, and the rightmost dial would store how many 1's there are. If we have a single dial, we could represent $10^1 = 10$ different values, with two dial we could represent $10^2 = 100$, with three dials $10^3 = 1000$ etc. In the general case, if we have n dials, we can represent 10^n different values.

1.3.2 Bits

Digital systems do not use decimal numbers because they are usually electronic systems, and in order to represent the ten different values of a decimal digit they would need to deal with ten different electric voltages. Technically this would be very difficult. As a consequence, digital systems use binary digits which have only two possible values: 0 and 1. The word "binary" comes from the Latin word for "two". This is technically much easier because the systems need to deal with only two different voltages: a very low voltage for 0, and a much higher voltage for 1. This is similar to a light switch: either the light is switched off (0) and no electric current is flowing, or it is on (1) and the electric current can flow and power the light. A binary digit is called a *bit* (short for *b*inary dig*it*).

If we want to store bits, we could do so with switches. A switch has two different states, on and off, which correspond to the binary values 0 and 1. Of course, in an electronic system the switches would not be mechanical but some sort of electronic switching units. For each bit that we want to store, we install a switch. With a single switch, we could represent $2^1 = 2$ different values. With two switches we could represent twice as many values, that is $2^2 = 4$: for each of the two states of the first switch the second switch could assume two different states. If we add a third switch, the number of possible values again doubles to $2^3 = 8$ etc. Note that this is just like the aforementioned dials we used to explain storage of decimal numbers, only that the binary system is based on the number 2, so that with n bits (or switches) we can represent 2^n different values.

1.3.3 Converting Binary to Decimal Numbers

When we read numbers, we go through the digits and because we know the quantity each digit represents we can understand what quantity is represented by the number as a whole. In this regard, binary numbers are just the same as decimal numbers. Let us first have a look at how we actually understand the quantity represented by a decimal number:

$$\begin{aligned} 135 &= 1 \times 100 + 3 \times 10 + 5 \times 1 \\ &= 1 \times 10^2 + 3 \times 10^1 + 5 \times 10^0 \end{aligned}$$

The number "one hundred and thirty-five" is made up of one 100, three 10s and five 1s. Going through the decimal digits from right to left, the rightmost digit always refers to the 1s, i.e. the quantity $10^0 = 1$. The next digit refers to the 10s, i.e. $10^1 = 10$. In general, the ith digit from the right refers to the quantity 10^{i-1}. The value of the whole number can always be calculated arithmetically by multiplying each digit with the quantity it refers to, and then adding up all these products.

Let us now consider how we can calculate the decimal value of a binary number. The key is to know which quantity (as a decimal number) each of the bits in the binary

number represents. Let us consider the binary number 10101_b (note that in the following equation we add the suffix $_b$ to all binary numbers to distinguish them clearly from the decimal numbers):

$$\begin{aligned} 10101_b &= 1\times 16 + 0\times 8 + 1\times 4 + 0\times 2 + 1\times 1 \\ &= 1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0 \\ &= 21 \end{aligned}$$

Going through the bits from right to left, the rightmost bit always refers to the 1s, i.e. the quantity $2^0 = 1$. The next bit refers to the 2s, i.e. $2^1 = 2$. The bit after that refers to the 4s, i.e. $2^2 = 4$, etc. In general, the ith bit from the right refers to the quantity 2^{i-1}. Just as for the decimal numbers, the decimal value of the whole binary number can always be calculated arithmetically by multiplying each digit with the decimal quantity it refers to, and then adding up all these products.

1.3.4 Bytes

A single bit can only represent two different values, which is not much. In order to store information such as text or images it makes sense to combine the bits into groups, so that more values can be represented. A group of eight bits is called a *byte*. From the previous sections we know that a byte can be used to represent $2^8 = 256$ different values.

A common use of a byte is to store a single character. A common way of encoding characters in a single byte is called ASCII (pronounced “ask-ee”, short for American Standard Code for Information Interchange). ASCII simply defines a mapping between each of the numbers of a byte from 0 to 255 to a character. For example, the character “a” has the number 97.

When describing the size of data, we usually use bytes as a unit. For example, if we have a text document on our computer that contains 50 characters in ASCII encoding, then the size of this document would be 50 B (“B” is the common abbreviation for “bytes”). Many types of data are thousands, millions or even billions of bytes big. As a result, we use prefixes in front of the B that express that we mean thousands, millions or billions etc. of bytes. The prefixes are the same as the ones used for meters, e.g. thousand meters are one kilometer ($1000m = 1km$) and thousand bytes are one kilobyte ($1000B = 1kB$). The following table lists the most important prefixes, and also points out what types of data usually require numbers of bytes in the order of a prefix.

Prefix	**pronounced as**	**is equal to**	**used for**
kB	kilobyte	1,000 B	text documents small and medium sized images
MB	megabyte	1,000,000 B	large images music files (e.g. MP3 files)
GB	gigabyte	1,000,000,000 B	movies (e.g. on a DVD)

1.4 Recommended Reading

- Digital systems

  - http://en.wikipedia.org/wiki/Digital

- Binary
  - http://en.wikipedia.org/wiki/Binary_numeral_system
- Bits
  - http://en.wikipedia.org/wiki/Bit
- Bytes
  - http://en.wikipedia.org/wiki/Byte
- Standard decimal prefixes
  - http://en.wikipedia.org/wiki/SI_prefix
- Binary Prefixes
  - http://en.wikipedia.org/wiki/Binary_prefix

1.5 Self-Test Questions

1. What is the main difference between a digital and an analogue system?
2. What is sampling? Give a definition and an example.
3. How can we encode the colours in a picture?
4. How many different values can we represent with 7 bits?
5. Which decimal number corresponds to the binary number 10111?

CHAPTER 2

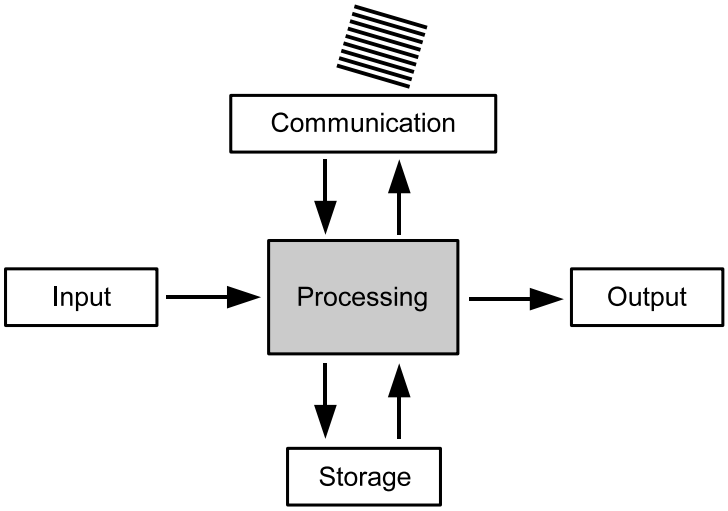
Computer Systems

A computer system consists of both hardware and software. Computer hardware is the physical equipment that makes up a computer system. Some people call this “the stuff you can kick”. In this chapter, we will discuss what the most important pieces of hardware do.

2.1 Introduction

A computer is an electronic machine that can automatically execute simple instructions. Many of the instructions tell the computer to manipulate data, or move them from one place in the computer to another. For example, a computer can add, subtract, multiply and divide numbers. It can also compare them and make simple decisions based on comparisons such as “number x is equal to zero” or “number x is greater than number y”. Usually computers can receive new data from users, e.g. through a keyboard, and they can also show their data to users, e.g. on a screen.

The main part of a computer is called the central processing unit (CPU), which is the part that executes the instructions given to the computer. However, a number of additional components are required so that we can use computer systems effectively. These components perform jobs such as translating information between humans and computers (input and output), storing information, and transferring information between different computers.

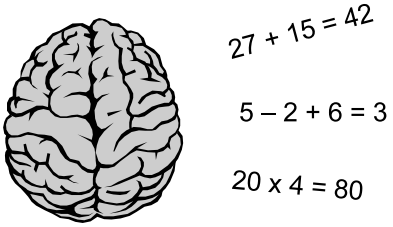


Components of a typical computer system

2.2 Processing hardware

The most important part of any computer system is the Central Processing Unit (CPU). It reads and follows the instructions that make up a program. It also does any calculations required, and controls the rest of the computer system.

> *Analogy: It might help to think of a CPU as the "brain" of the computer. Like a brain, the CPU receives input from input devices, processes and makes decisions based on that data, and sends commands to output devices. In the case of a computer, common input devices are the keyboard and the mouse, and common output devices are a screen, speakers or a printer. The "input devices" of our brain are our sensory organs, i.e. our eyes, ears etc. The brain's "output devices" include our voice and the skeletal muscles that allow us to move freely. However, keep in mind that computers are not intelligent, in contrast to us human beings. CPUs are very good at performing exact operations such as adding or comparing numbers, but if we do not tell them exactly what to do they are completely useless. They cannot "think" by themselves.*



The performance of a CPU is commonly measured in operations per second. How many operations a CPU can perform in a second depends on several factors. One factor is the CPU's *clock speed*, i.e. the frequency with which electrical impulses are sent through

the CPU. The more electrical impulses per second we send through a CPU, the more operations it will be able to perform.

It is worth noting that while the CPU can follow instructions extremely rapidly, it is limited by the speed of the other components. If a job requires a lot of calculations, then the speed of the CPU will be the most significant factor. However, if the job relies heavily on other parts of the computer, the speed of the CPU may not be important because the other parts may act like bottlenecks that slow down the whole process.

> *Analogy: For example, you might take an hour to make a decision about the colour to paint a room in your house. It will take substantially longer to actually paint the room, perhaps as long as a week. In this case, making the decision twice as fast will not make any real difference to the time it takes for the room to be painted.*
>
> *However, if you were buying a car then it might take an entire week to decide exactly what model of car you wanted. Paying for the car would only take a few minutes. In this case, making the decision twice as fast makes a significant different to the time it takes to buy a car.*

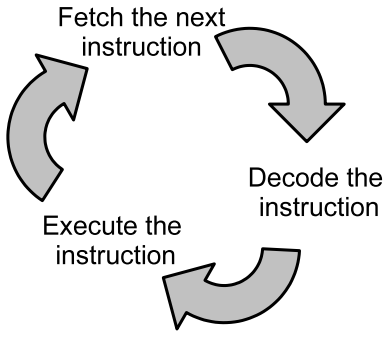
2.2.1 Inside a CPU — Advanced, not examinable

Although there are many different designs of CPU, they all have some common features. These are discussed below:

Main Control Unit

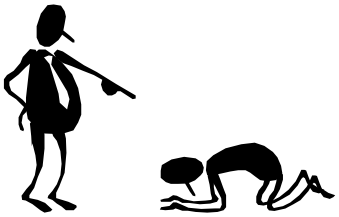
The main control unit is the part of a CPU that controls everything. The main control unit follows a repetitive cycle where it

- asks for the next instruction
- decodes that instruction (figures out what needs to be done), and
- executes the instruction (tells the other parts of the system what to do so that the instruction is followed).



The fetch-decode-execute cycle of a main control unit

Analogy: It may help to think of the main control unit as the manager of the CPU. It is responsible for figuring out what to do next and ensuring that it actually gets done. If we think of the CPU as being a large building, then the main control unit would be the manager in charge of that building.



Arithmetic and Logic Unit

The arithmetic and logic unit is designed specifically to do calculations. The circuits in this unit can perform simple arithmetic on numbers. It holds the logic circuits that allow the computer to compare numbers to see if they are equal to, greater than, or less than one another. It can also apply logical operators such as AND, OR and NOT.

Analogy: It might help to think of the ALU as a mathematician that sits in an office doing calculations all day. People bring numbers to the door and ask the mathematician to do something with them (add, subtract, multiple, divide). The mathematician can quickly do the calculation and pass the result back.

Registers

The CPU needs a place that it can hold the numbers that it is currently working on. The registers provide this temporary storage location. Each register can hold a single number. Each CPU has a very small number of registers (typically less than 100).

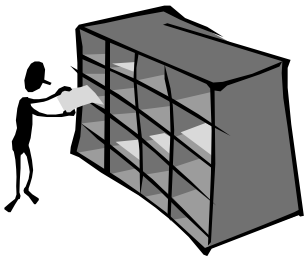
For example, if an instruction tell the computer to add together the numbers 7 and 8, then the CPU needs to remember the number 7, the number 8 and also the result of adding the numbers together, the number 15.

Analogy: If we think of a mathematician in an office being the ALU, it might be useful to think of the registers as being boxes that sit on the shelf of the office. The mathematician can quickly do many complicated calculations by getting numbers from the boxes and putting any answers into other boxes.

Cache

Accessing the information in a register is extremely rapid, but there are very few registers available. Getting any information from outside the CPU takes a long time, since the information has to travel a long way to get from the primary memory to the CPU. Most modern CPUs have a small amount of additional memory that is included inside the CPU itself and therefore is able to be accessed fairly rapidly (although not as fast as the registers). This is called the cache.

> *Analogy: Returning to our mathematician, the cache would be like having a set of storage lockers located in the same building as the mathematician. We might have thousands of storage lockers in the basement. It is reasonably fast to get information out of those lockers, but if were working on the numbers regularly, we would prefer that they were stored in the registers. However, we only have a small number of registers, so numbers that we don't need as often would have to be put into the lockers in the basement.*



Bus

A bus is an electronic pathway that is used to transfer information. It carries information from one part of the computer to another. There are normally a few different buses inside a CPU. The size of the bus determines how much information can be transferred at any one time.

Analogy: We return again to our mathematician in their office. Imagine that they have a small lift that runs through the floor, down into the basement where the lockers are stored. When the manager (CPU) gets an instruction telling them to multiply the number stored inside locker 7829 by the number stored inside locker 998, then they would send send a request out for the numbers to be delivered. A person in the basement would open up the right lockers and get out the right numbers. They would ride up in the lift and put the numbers in the correct boxes in the mathematician's office. We can think of the lift as being the bus.

2.3 Storage components

Computers store a lot of information. This information can be stored in different ways, using different pieces of hardware. The devices that store information for long-term use are generally very cheap, but slow. We call these devices *mass storage* or *secondary storage* devices. Information that needs to be accessed very rapidly has to be stored in more expensive components known as *primary memory*.

2.3.1 Primary memory

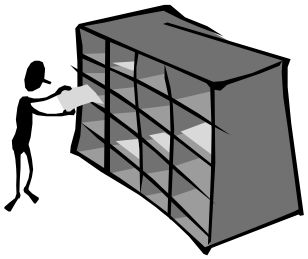
The form of primary memory that we are most concerned with is the Random Access Memory (RAM). This is the memory that stores any programs that we are currently executing, and the data that we are working on.

Because the CPU can follow instructions very rapidly, it has to be able to retrieve the next instruction from memory as fast as possible. When we want to run a program (i.e. a series of instructions), we copy those instructions from a mass storage device (such as a hard drive) into the RAM. When the CPU asks for the next instruction, it is loaded from the RAM into the CPU where it can be decoded and executed. Any data that is required by the CPU is also shifted from the mass storage device into the RAM. That way, when the CPU needs to read some data, or make changes to the data, it will happen very rapidly.

RAM consists of electronic circuits that store an electrical charge. These circuits are normally printed onto silicon chips. Memory of this type is extremely fast to access, since it only requires an electrical charge to be sent from one location to another. Manufacturing memory of this type is fairly expensive (usually about 100 times as expensive as mass storage devices). Although the RAM is extremely fast to access and modify, when the power is turned off, the information is lost. This kind of memory is known as *volatile* memory.

Analogy: Think of the CPU as a building filled with people doing paperwork. The instructions that specify the jobs that need to be done are located at a nearby warehouse. An instruction is fetched from the warehouse and brought back to the CPU building where it is executed. Once the instruction has been completed, the next instruction is fetched from the warehouse. These instructions are very simple, so they might say something like "Add the number located in locker number 77445 to the number stored in locker number 982 and put the answer into locker 14528".

In this analogy, the warehouse plays the role of the RAM.



The physical RAM is divided up into a set of discrete boxes, each of which can hold a single byte (a number between 0 and 255). The number of boxes in a normal RAM chip is huge. A RAM chip that holds 1 GB of memory will have space to store 1,000,000,000 different numbers.

2.3.2 Mass storage (Secondary storage)

Devices used for mass storage are capable of storing information over a long period of time while the computer is switched off. Accessing this information is much slower than accessing RAM (thousands of times slower). It is much cheaper to store information using a mass storage device than it is to store it on RAM (hundreds of times cheaper). When we compare mass storage devices with primary memory, we see that mass storage devices are:

- cheaper
- slower
- and store information when the power is turned off (non-volatile).

The most commonly used forms of mass storage are: hard disk, solid-state drive (SSD), optical disk (CD-ROM / DVD). Magnetic tape is still used for backup as it is very cost effective. However, access is very slow compared to the other storage devices.

2.4 Input components

Input devices are those components that are used by humans to provide information to the computer. These devices are used to put data into the computer, hence the term "input". This type of equipment often acts as a translator, converting signals which are used and understood by humans into an electronic form which can be processed by a computer. The most common device of this type is a keyboard, which allows a user to

perform almost any task. Other input devices are typically used for more specialised tasks.

A graphical user interface often requires a device to control a pointer, and the most common device for this task is the mouse. However, other devices such as the light-pen, tablet, and track-ball provide more flexibility for specialised applications. Today, touch screens are becoming more common in publicly accessible terminals where another form of pointing device is likely to be damaged.

Direct input is required where the data is too complex (or it is inconvenient) to be entered using a keyboard or pointing device. An image scanner or fax machine is used to input entire images into the computer, and bar-code readers and magnetic stripe readers provide a quick (and private) method to enter specific information. Optical recognition systems are used by banks to read cheque numbers (using magnetic-ink character recognition), and by other organisations to read pen or pencil marks in allocated spaces (e.g. lotto sheets, or multi-choice examinations). More recently, the development of software has encouraged voice input (using a microphone) as a method of dictation or control of computer systems.

2.5 Output components

Output devices are the complement to input devices. Equipment used for output acts as a translator, converting the digital signals a computer uses into a form which is readily understood by humans. The monitor or screen is the most common example of an output device. Any device that produces something understandable (to humans) from the computer is classified as an output device. Common examples include screens (monitor), speakers, printers and plotters.

CHAPTER 3

Online Publishing

3.1 Introduction

The Internet is a communication medium that we can use to publish information and distribute our ideas. There are commonly available tools that provide a way for us to easily publish information alone, or work collaboratively with others.

3.2 The World-Wide Web as a media source

The World-Wide Web has provided the opportunity for anyone to publish anything. However, although it is *possible* for anyone to publish, a certain amount of technical expertise is required to create and maintain a web site. The difficulty in authoring and hosting a web site is a barrier to most people.

It is also difficult to contribute to a community through the publication of web sites since each web site exists independently. It is easy to author a site which links to other content, but getting reciprocal links to form a network of related information can be difficult or impossible.

The benefits are that you own and control the content on your own web site and can express any opinion you have. The costs are that each web site is independent, rather than part of an existing network of pages. It can be technically difficult, financially costly, impossible to integrate into an existing community. Few people may ever find or read the web site.

Most content on the web is produced by individuals, or by small teams of people who know each other. A single person will typically write the content. This content will often be reviewed and edited before it is published. Small teams of people are sometimes

given the responsibility of writing or maintaining content about a particular topic. These individuals or teams that publish content are frequently normal members of the public without any official affiliation with a traditional media organisation.

3.3 Blog

The word “blog” was shortened from the term “weblog”. A weblog is a web page which consists of a series of posted messages. These messages normally appear in reverse chronological order (i.e. with the newest message appearing at the top of the page). Blogs are frequently used to maintain online journals. The content published in blogs is often the opinion of an individual (similar to the editorial in a newspaper).

3.4 Wiki

A Wiki is a piece of server software that allows users to freely create and edit Web page content using any Web browser. Wiki supports hyperlinks and has a simple text syntax for creating new pages and crosslinks between internal pages on the fly.

The content stored by a wiki is produced collaboratively by the users of the system.

The wiki system is claimed to work because:

- Everybody feels that they have a sense of responsibility because anybody can contribute.
- Any information can be changed or deleted by anyone. Wiki pages represent consensus because it’s much easier to delete insults than to indulge them. The content that remains is naturally meaningful.
- Anyone can play. This sounds like a recipe for disaster, but to make an impact on a Wiki, you need to generate real content. Anything else will be removed. So anyone can play, but only good players remain.
- There’s usually a strong commitment from the wiki community to keep the wiki clean and nice. Everyone uses it, so they all try to maintain it in a usable state.
- A wiki is not like online chat. It doesn’t work in real time. People take time to think, sometimes days or weeks, before they follow up some edit. So what people write is generally well-considered.

3.5 Recommended Reading

- Blogs
  - http://en.wikipedia.org/wiki/Blog
- Wikis
  - http://en.wikipedia.org/wiki/Wiki

CHAPTER 4

Wiki

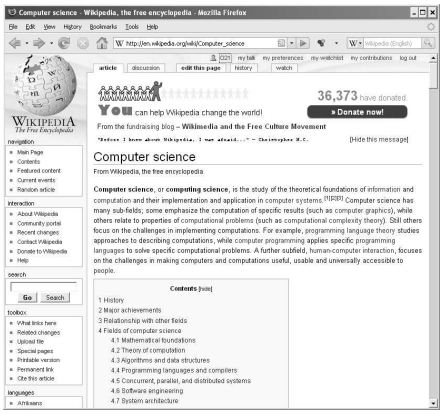
4.1 Introduction - What is a wiki?

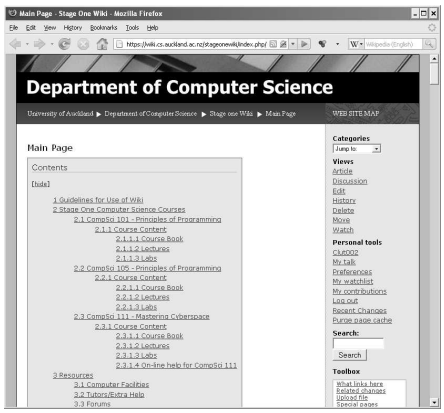
A *wiki* is a system that allows a user to easily create and edit web pages. However, unlike traditional web pages, the content in a wiki is able to be edited by *anyone*. Anyone using the system can freely edit the content of the pages, add hyperlinks and rearrange the structure of the pages as they wish. Since a wiki is designed to be extremely easy to use and freely accessible to everyone, it provides a real opportunity for anyone who wants to contribute. This "open-editing" approach encourages the democratic use of the web and promotes the development of content by non-technical users.

4.2 The stage one wiki

We have set up a wiki for stage one students in Computer Science. Everyone enrolled in a stage one Computer Science course will have an account set up for them so that they can log in and start contributing. Note that the Computer Science wiki uses the same software as the Wikipedia, i.e. the MediaWiki system, but looks a bit different because it was changed to fit better into the Computer Science website. The screenshots and descriptions in this chapter refer to the standard MediaWiki user interface, so they will differ a bit from the Computer Science wiki. However, the underlying software is the same, therefore you will find the same functions in the Computer Science wiki, although possibly at slightly different locations in the user interface. The figure below shows a Wikipedia page (left) and a CS Wiki page (right). In both wikis, the main document area works just the same. But while the Wikipedia arranges the links for the functions (e.g. "discussion", "preferences" and "search") at the top and left border of each wiki page, the CS wiki arranges those links at the right border. Some of the links are different, too:

for example, only the Wikipedia offers a link “Donate to Wikipedia” and the possibility to change between different languages.





4.2.1 Teaching and learning

Students learn best when they are involved in activities. The process of actually doing something engages more attention and promotes deeper understanding of the material. Research has shown that one of the best ways to learn something is to try and teach it to someone else. This applies to students of all ages, from primary school through to post-graduate students.

In order to explain a topic to somebody, information must be arranged so that it is easy to understand. The act of structuring the material requires the teacher to really think carefully about the topic, and they almost always learn something from the process. Because students learn so much when they try and teach, a really good learning environment will provide the opportunity for students to teach others. Using a wiki will help us achieve this goal.

4.2.2 Expectations

Everyone involved in the course is expected to contribute regularly to the Computer Science stage one wiki. The more that a wiki is used, the more useful it becomes. Everyone can contribute something, and anything that is contributed is likely to be edited and improved over time. Some of the ways you might be able to contribute are:

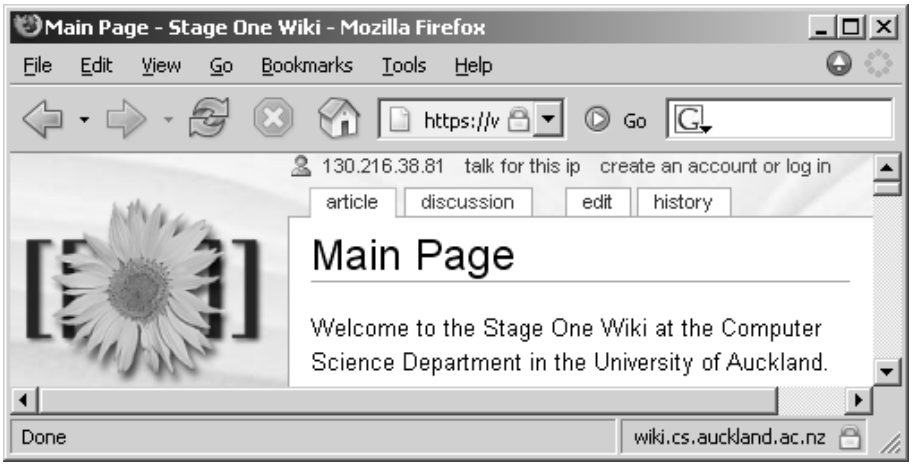
- Writing a new page about a topic
- Improving an existing page
- Correcting spelling and grammar
- Rearranging the structure of the information (reorganising within a page, or reorganising the pages themselves
- Creating a Frequently Asked Questions list (FAQ)
- Asking questions

- Answering questions on a Frequently Asked Questions list (FAQ)
- Maintaining a table of contents or index of important topics
- Creating or adding to a set of links to other helpful resources
- Contributing to a discussion about one of the pages

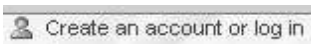
If you want a page on a particular topic, then write one yourself (or at least make a start). Once a page is created then other students will probably start to add content and improve the page. Always remember that a wiki is designed to be a tool that allows a community of people to work together to create content.

4.3 Using MediaWiki

MediaWiki is the product used to create Wikipedia. It is one of many wiki systems currently available. MediaWiki is distributed under the GNU General Public Licence, so that means that it is free and open-source. It is reasonably easy to use and has a professional looking interface. We will be using MediaWiki throughout the course. When you arrive at the main page of a MediaWiki site, it will look something like the following:

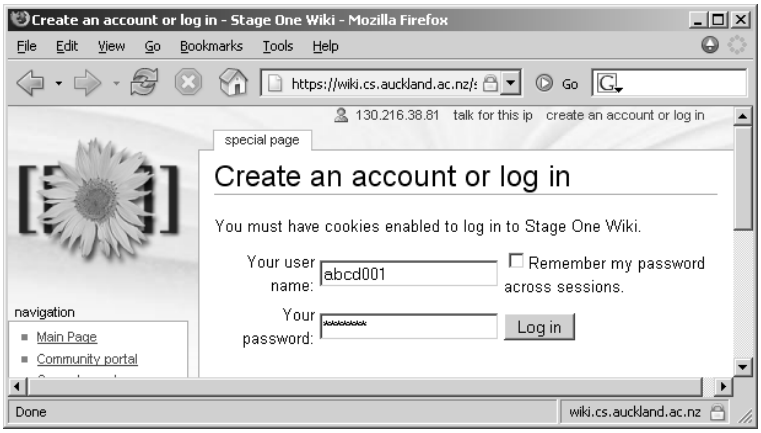


At the very top of the page, we see a link that allows us to “Create an account or log in”.

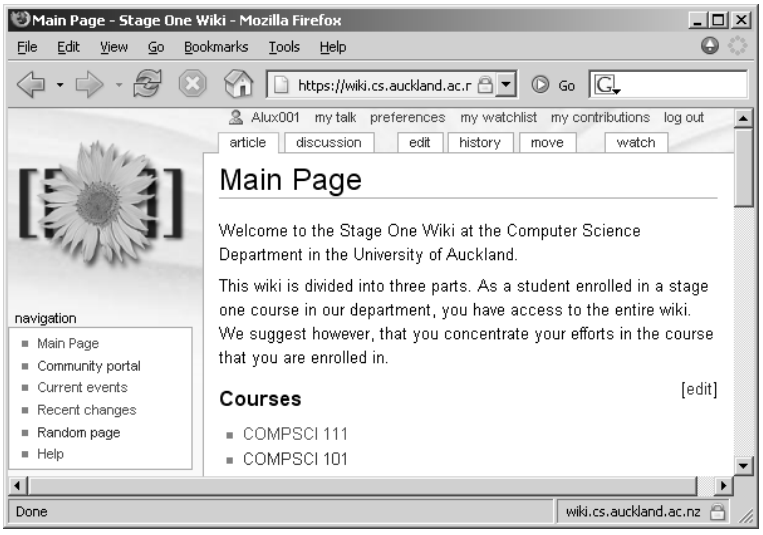


4.3.1 Logging in

MediaWiki has the ability to protect pages and restrict access. Anyone can read the wiki, but it can be set up to ensure that only some people can edit the pages. We have set up MediaWiki so that only students enrolled in the course can make changes. Before we can start contributing to the wiki, we must first log in. We use our normal NetLogin username and password here.



Once we are logged in, we can return to the main page and explore the system.

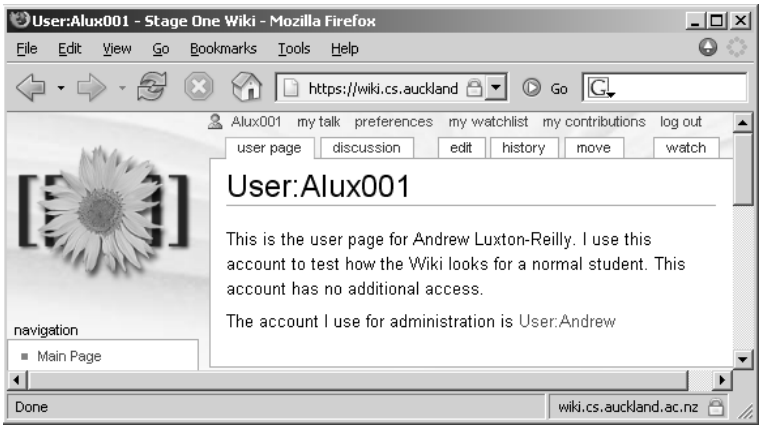


At the very top of the page, we now see a variety of links. They are (in order):

- User (alux001 in the screenshot above)
- my talk
- preferences
- my watchlist
- my contributions
- log out

user

The first link shows the name of the user. Each user of a wiki is given their own personal home page. Clicking on the name of the user will lead to the user's home page (within the wiki). Initially this will be empty, but each user should endeavour to include some simple details about themselves on their home page.

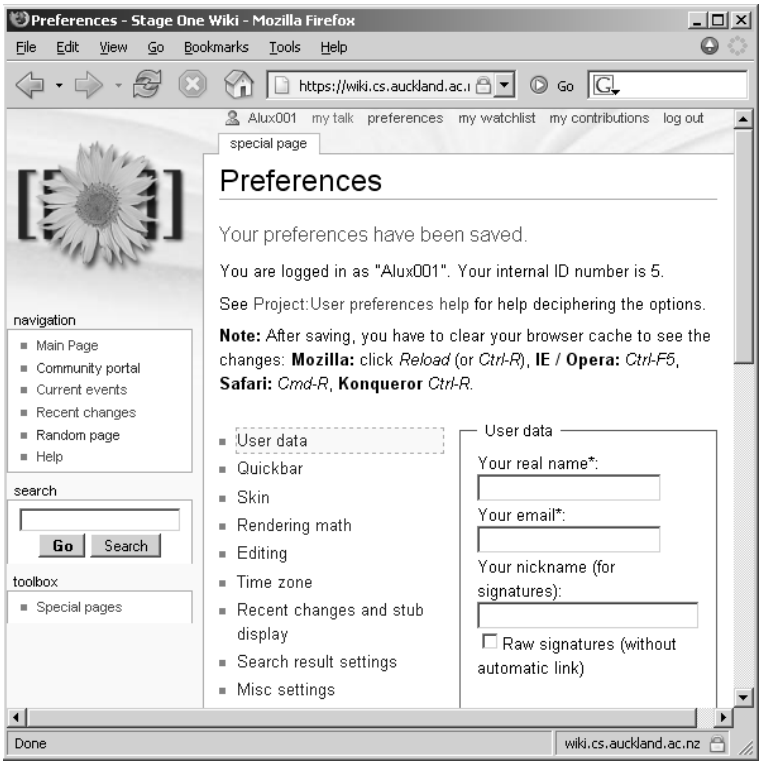


my talk

The second link leads to a discussion page about the user. Any discussion or personal messages directed towards an individual user should be posted in their discussion page.

preferences

The third link allows a user to customise the way the wiki displays information. Users are welcome to change these preferences.

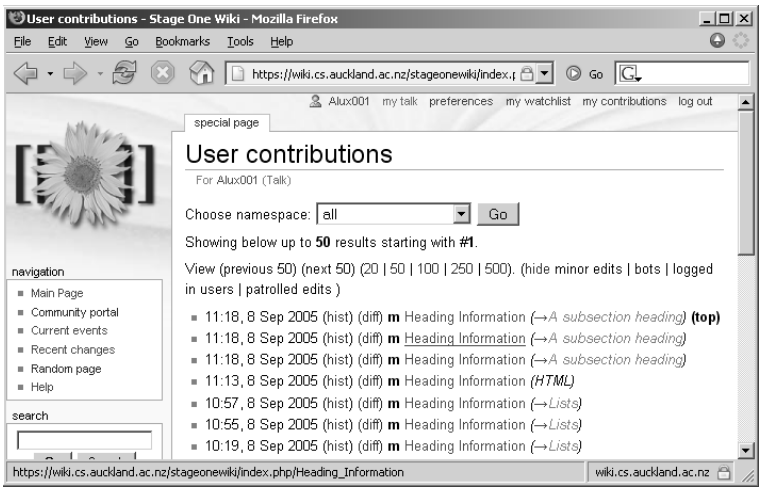


my watchlist

The fourth link leads to a page containing the watchlist of a user. This is a list of all the pages that a user has specified that they are interested in watching. The pages on the watchlist are listed in the order that they were last modified. Any pages that have been modified since the user last looked at them are highlighted. This makes it very easy to keep track of specific articles or pages that the user finds interesting. See section 4.3.9 for details about how to add a page to the watchlist.

my contributions

The fifth link displays a list of all the contributions the user has made to the content of the wiki.



log out

When we have finished with the wiki, then we should log out.

4.3.2 Tabs

Along the top of the page we can see a number of tabs. These are article, discussion, edit, history, move and watch/unwatch.

4.3.3 Article tab

The article tab contains the main content on this topic. Most users are interested in the articles that are contributed to a wiki. Anyone can read the articles contained in the wiki, even if they are not logged in.

4.3.4 Discussion tab

The discussion tab will display a page containing a discussion about the article. This is an area where users can make comments, suggest improvements, argue and generally discuss the topic in question. When a new article is created, there will be no actual comments on the discussion section. Whenever we have any questions or comments about a given article, then the discussion page is the ideal place to post them.

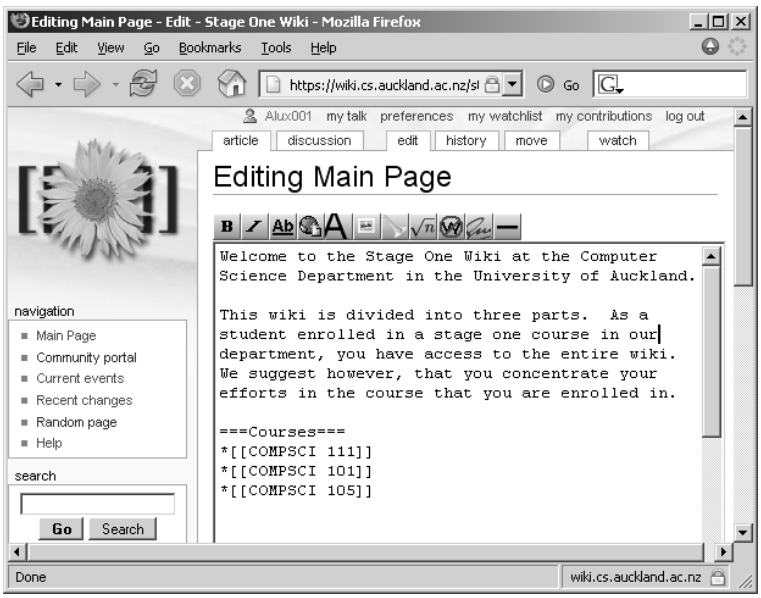
When a question or comment is posted in a discussion page, it is conventional to state who is making the comment or question. We should always include our name when we contribute to a discussion. See section 4.4.7 for more information about adding your name to a contribution. Note that you must be logged in to edit a discussion page.

4.3.5 Edit tab

The edit tab allows a user to edit the article shown on this page. We have set up the MediaWiki so that you must be logged in before you can edit a page. Not all wikis have this restriction. In fact, most wikis allow anonymous users to edit pages without logging in to anything.

Editing a page

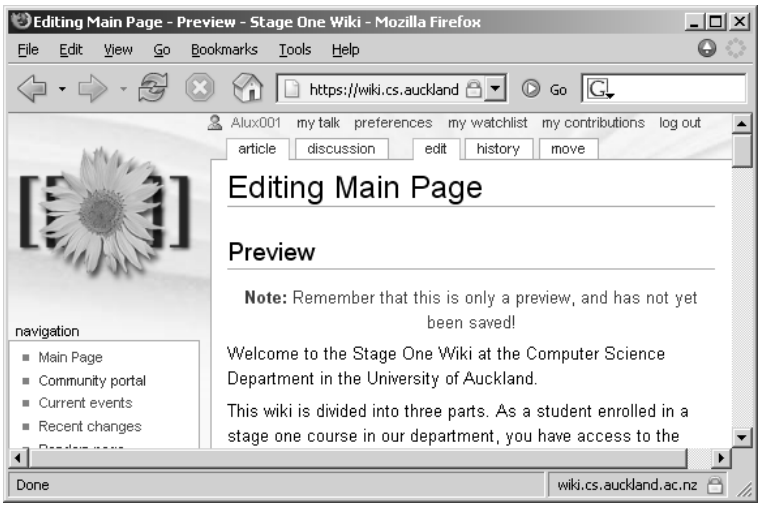
To edit a page we simply need to click on the edit tab which appears at the top of the page. This will allow us to immediately start making changes to the content.



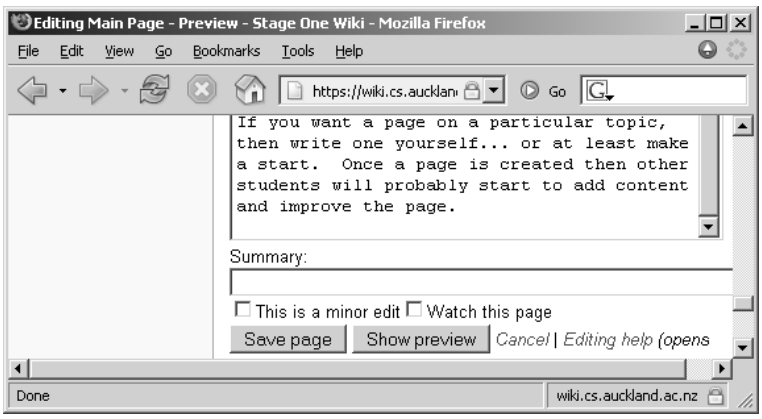
One of the advantages of using a wiki is that it is easy to edit pages. If we change a page and then realise we made a mistake, then we can easily edit the same page and correct the mistake.

Preview and save

When we are happy with the changes we have made, the Show preview button will allow us to check how it looks.

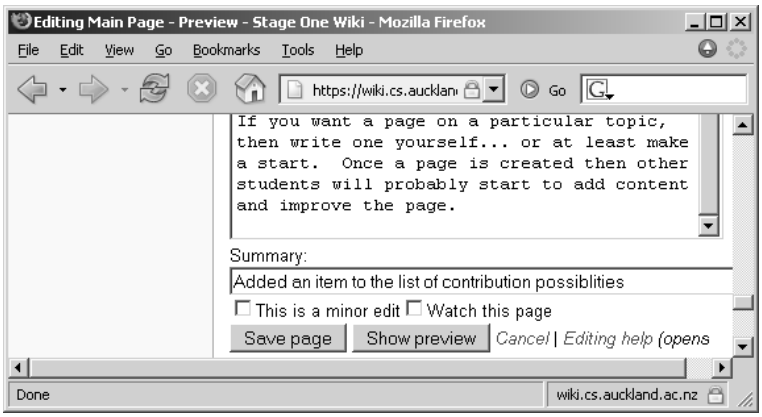


It is always a good idea to preview the changes and just verify that the page is correct before we commit to saving the page. We must remember to click the Save page button when we are finished.



Summaries

Note that there is a space to enter a summary of our changes. The summary should be used to describe the kind of change we made to the content. The purpose of providing a summary is so that people that are keeping track of the wiki can see what changes are being made without looking up the individual pages.

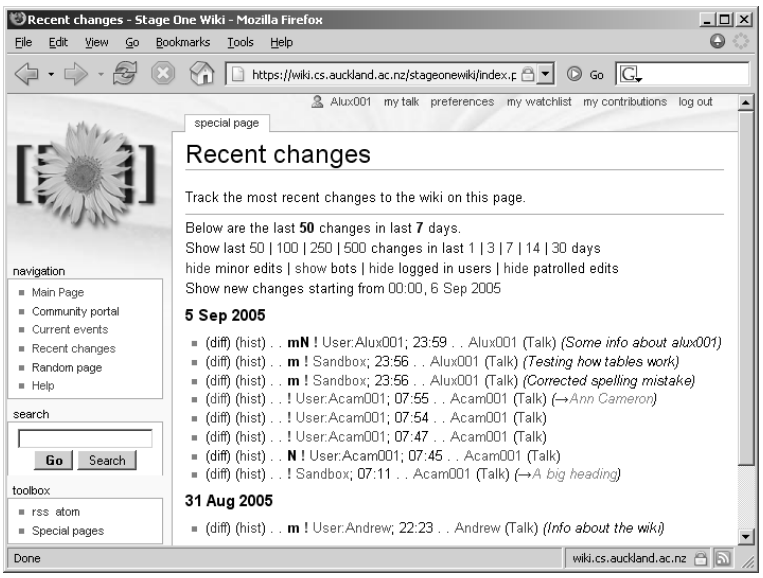


Some examples of suitable summaries might be:

- Corrected spelling mistakes
- Added section on variable identifiers
- Deleted paragraph on loops
- Updated links
- Added new FAQ item

Keeping up to date with changes

When we look at the recent changes to the wiki (by clicking on the link to recent changes in the toolbox that appears on the left side of the page) then you will see a list of changes along with their corresponding summaries.

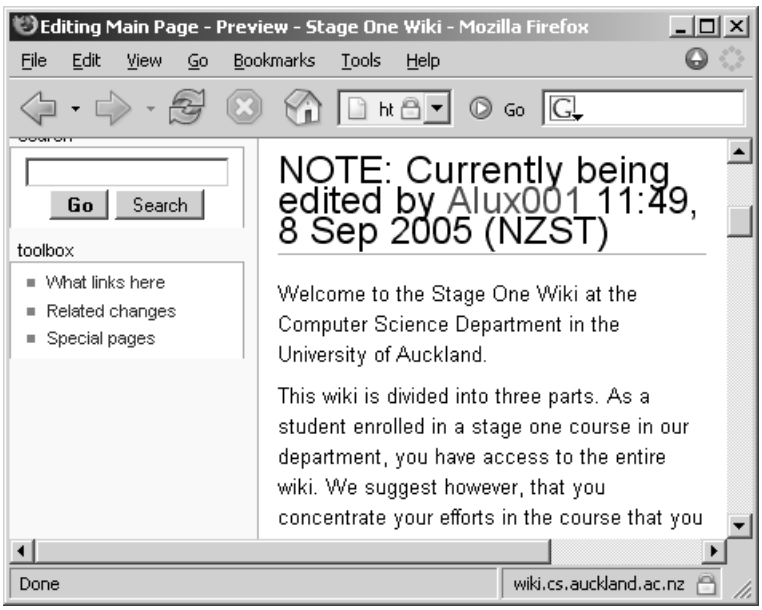


4.3.6 Editing conflicts

Since the content can be changed by anyone, it is possible that two people will try to edit the same page at the same time. This is especially common with popular pages, or pages that are changed frequently. If a lot of people are working on the same content at same time, then it can be useful to follow some simple conventions. These are outlined below:

Announcing your edits

When you are about to edit a page that you know is reasonably popular and might be changed by someone else, then you can simply add a note to the top of that page that tells people that you are working on it. For example:



If you are announcing that you are editing a page and you don't want other people to edit until you are finished, then it is a really good idea to include the automatically generated author link with the date and time included (see section 4.4.7 for more details). For example, the message in the example above was generated with the markup:

Page Source

```
1 == Note:  Currently being edited by ~~~~ ==
```

Removing your announcement

It is extremely important that you remember to remove any edit warnings when you have finished editing the page. If you say that you are working on a page, then you should work on it quickly, make the changes you want and remove the warning when you are finished.

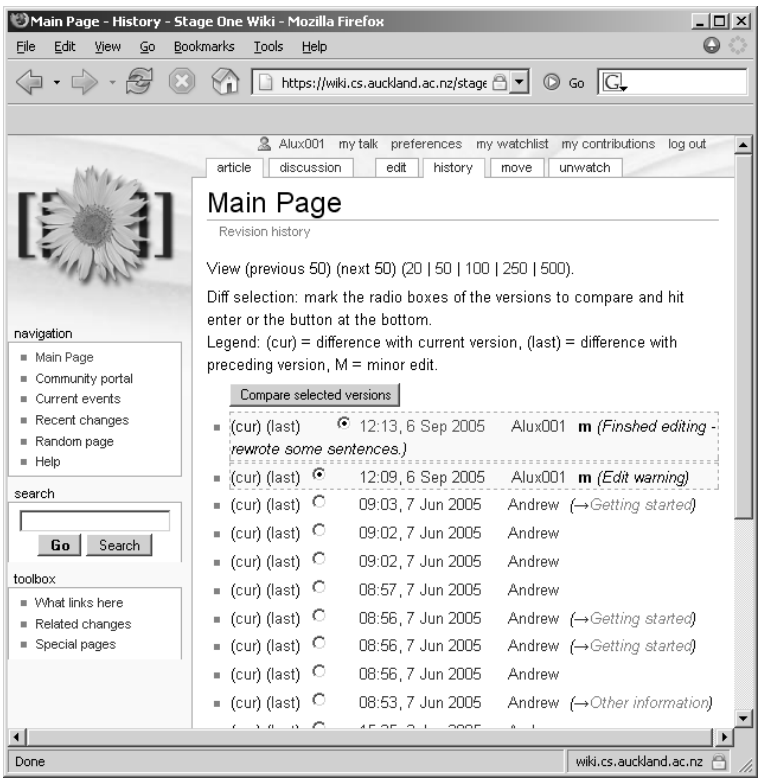
Saving your work locally

Whenever you fill in a form and submit it to a website, there is a chance that a problem will occur with the transfer and you will lose the information in the form. To prevent the loss of information in this way, it is always worthwhile to store a local copy.

Before you hit the Submit button on a web-based form, select the text that you want to submit and copy it to the clipboard. If a problem occurs during the process of submitting, then at least you have a copy stored on the clipboard. This copy can either be pasted into a text file and saved on your disk (so that you can submit it later), or can simply be resubmitted again (using the form).

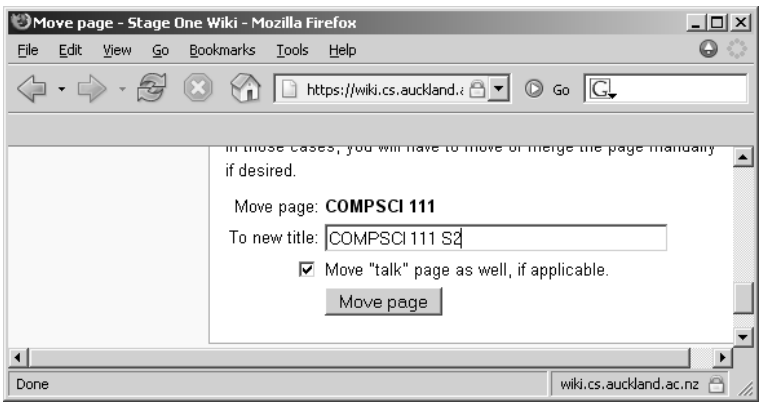
4.3.7 History tab

The history tab shows a list of all the changes made to that page. It is worth noting that *all* the previous changes that are made are recorded. It is easy for the community to see who is changing the pages, what they changed the page from, and what they changed the page to. This recorded history ensures that people are accountable for the changes that are made to the wiki. This in turn encourages high quality contributions and discourages destructive behaviour.



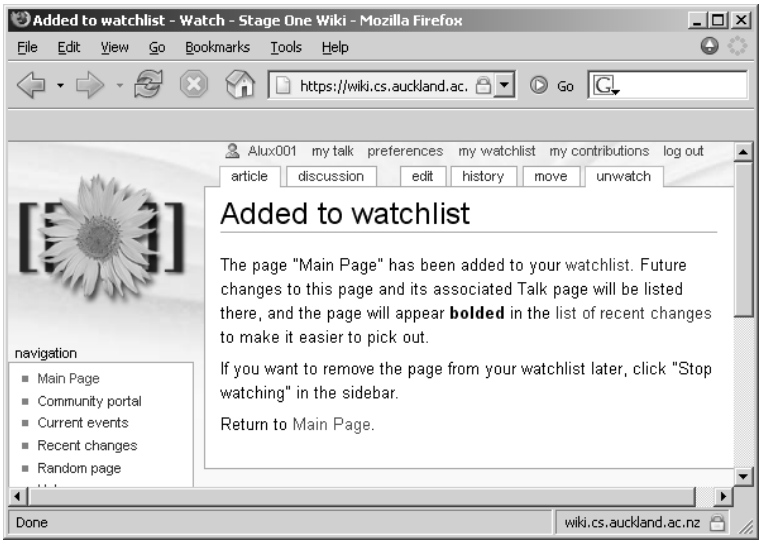
4.3.8 Move tab

The move tab allows us to rename a page. If the page is renamed, then any link that connected to that page will need to be updated. This is not recommended for popular pages that have a lot of links that lead to them. Renaming a page should be done only when necessary.



4.3.9 Watch tab

Clicking on the watch tab for a given page will add that page to our watchlist. We can easily keep track of changes that are made to that page.



4.4 Markup

The markup language that is used in a wiki is kept deliberately simple to ensure that people can learn to use the language quickly and easily.

4.4.1 Headings

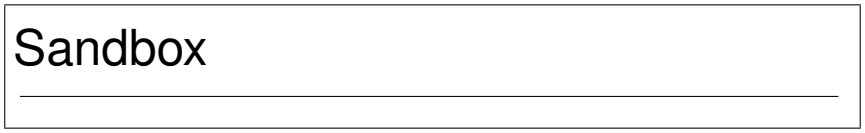
MediaWiki supports four different levels of heading. To mark some text as being a heading, a series of equals signs (=) are placed before and after the text. The following table summarises the different headings available.

Wiki Markup	Meaning
`= Text =`	Level 1 heading (most important)
`== Text ==`	Level 2 heading
`=== Text ===`	Level 3 heading
`==== Text ====`	Level 4 heading (least important)

Table 4.1: Headings supported by MediaWiki

Level 1 heading - main page heading

At the top of each page appears the main heading of that page. The main heading is a level 1 heading, and it is the same as the name of the page. For example, a page called "Sandbox" will begin with the word "Sandbox" in large typeface. A horizontal ruled line will appear below the main heading:



Each page should have only one main heading, so there is no need to create additional level 1 headings.

Level 2 heading - section heading

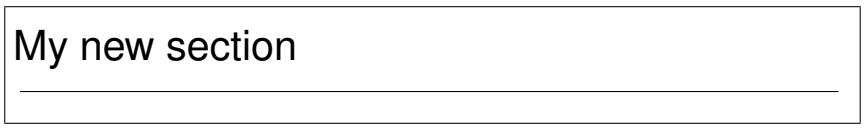
The largest heading size that we manually add to a page should be a level 2 heading (a section heading). Each new section we write should start with a new section heading.

The syntax for this is to add two equals signs either side of the section name as follows:

Page Source

```
1 ==My new section==
```

Each section appears in a large bold font. A horizontal ruled line appears below the section heading as follows:



Level 3 heading - subsection

To create a subsection, we use three equals signs either side of the subheading as follows:

Page Source

```
1 ===My new subsection===
```

This will appear on screen as a large bold heading, but not quite as large as the section heading. There will be no ruled line below a subheading.

My new subsection

Level 4 heading - sub-subsection

The lowest level of heading is used to create a sub-subsection. To create this level of heading, we use four equals signs either side of the name. For example, the markup:

Page Source

```
1 ====The smallest level of heading====
```

will result in the heading:

The smallest level of heading

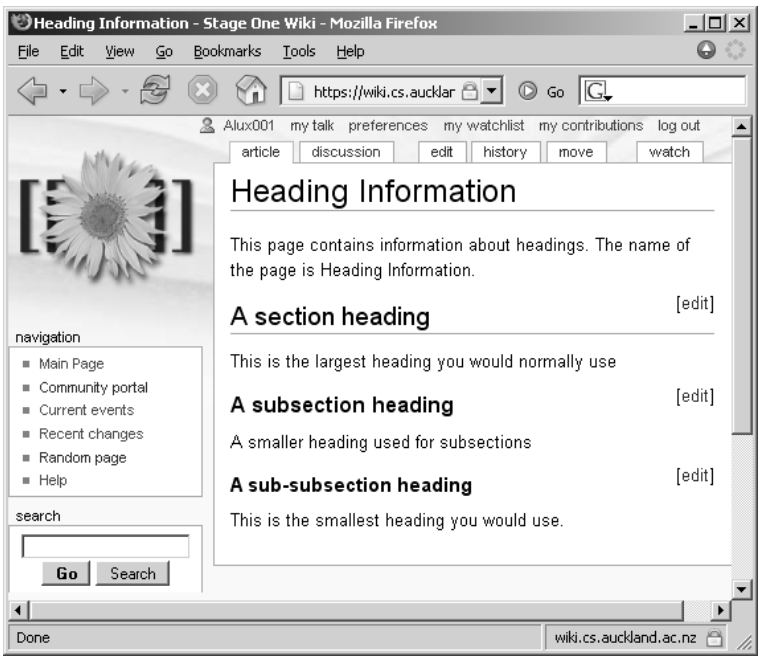
Example

The following page illustrates the use of different levels of heading:

Heading Information

```
1  This page contains information about headings.
2  The name of the page is Heading Information.
3
4  ==A section heading==
5  This is the largest heading you would normally use
6
7  ===A subsection heading===
8  A smaller heading used for subsections
9
10 ====A sub-subsection heading====
11 This is the smallest heading you would use.
```

When the page is displayed in a wiki, it will appear as follows:



4.4.2 New lines

To start a new paragraph, we need to leave a line empty. A single newline (e.g. created when you hit the Enter key) has no effect on the layout of the page.

For example, the text:

Page Source

```
1 This is part of a
2 sentence.  However, there
3 are many different lines here.
4
5 This is a new paragraph.
```

will be displayed in a wiki as follows:

This is part of a sentence. However, there are many different lines here.

This is a new paragraph.

4.4.3 Lists

Three kinds of lists can be created; ordered lists, unordered lists and definition lists. The following table summarizes the different lists available:

Each kind of list is described in more detail below.

Wiki Markup	Meaning
`#Item`	Ordered list
`*Item`	Unordered list
`;Term : Definition`	Definition list

Table 4.2: Lists supported by MediaWiki

Ordered lists

An ordered list is created by putting a hash sign (also known as the number sign) at the start of each line. For example, the wiki markup:

Page Source

```
#Apple
#Orange
#Banana
```

will be displayed in a wiki as follows:

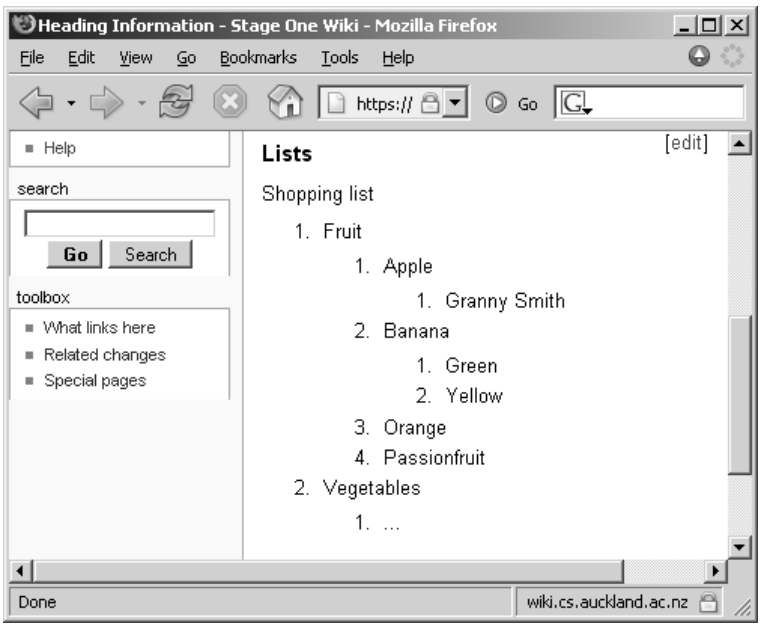
1. Apple
2. Orange
3. Banana

These lists can be nested by simply increasing the number of hash signs used at the start of each item. The more hash signs that are used, the greater the indentation level of the list. For example, the wiki markup:

Page Source

```
====Lists====

Shopping list

#Fruit
##Apple
###Granny Smith
##Banana
###Green
###Yellow
##Orange
##Passionfruit
#Vegetables
## ...
```

will be displayed in a wiki as follows:



Unordered lists

An unordered list is similar to an ordered list, except that instead of using a hash sign, an asterisk is used at the start of each line. For example, the wiki markup:

Page Source

```
1 *Apple
2 *Orange
3 *Banana
```

will be displayed in a wiki as follows:

- Apple
- Orange
- Banana

These lists can be nested in the same way that the ordered lists are nested.

Definition lists

A definition list is used to give a definition for a term. Firstly, the term needs to be listed, then the definition needs to be listed. The term is prefixed with a semicolon and the definition is prefixed with a colon. For example, the wiki markup:

Page Source

```
1 ;Apple : A brand of computer
2 ;Orange : A county in California
3 ;Banana : A fruit
```

will be displayed in a wiki as follows:

Apple
: A brand of computer
Orange
: A county in California
Banana
: A fruit

4.4.4 Indentation

Using a colon at the start of a line will indent that paragraph of text. For example, the wiki markup:

Page Source

```
This is a normal line of text.

: This line of text begins with a colon,
so the entire paragraph will be indented.
This can prove to be very useful when
some text needs to stand out from other
text (e.g. for examples, quotes, answers to
questions and so on).

This is another normal line of text.
```

This is a normal line of text.

: This line of text begins with a colon, so the entire paragraph will be indented. This can prove to be very useful when some text needs to stand out from other text (e.g. for examples, quotes, answers to questions and so on).

This is another normal line of text.

4.4.5 Pre-formatted text

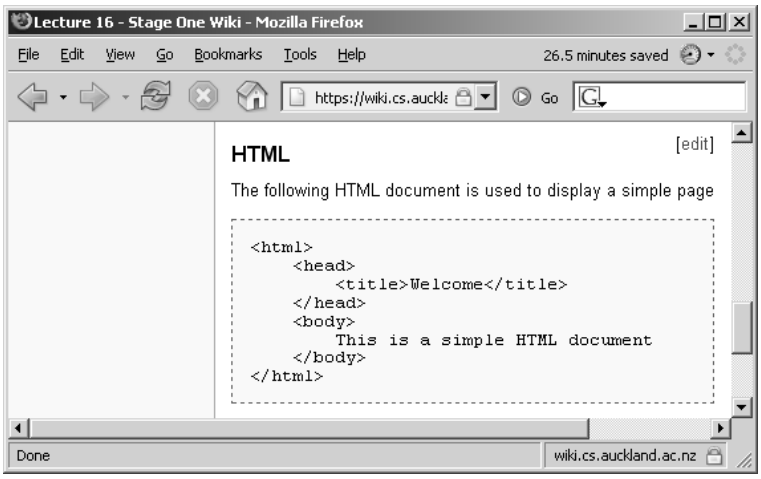
There are occasions when we want the text to remain exactly as we type it. A fixed-width font will be used to display such text, which is commonly called pre-formatted text. To ensure that the text is treated as pre-formatted text, we must start the line with a blank space. If any given line begins with a space, then the wiki will display that line of text as pre-formatted. A box will be drawn around the text and it will be separated from the surrounding text. This is very useful for displaying computer code.

For example, look very carefully at the following wiki markup. We have started all of the lines containing HTML code with a single space. This page:

Page Source

```
The following HTML document is used to display
a simple page
  <html>
     <head>
         <title>Welcome</title>
     </head>
     <body>
         This is a simple HTML document
     </body>
 </html>
```

will be displayed as follows:



4.4.6 Horizontal lines

To create a horizontal line, simply use four minus signs in a row `----`. For example, the wiki markup:

Page Source

```
Above the ruled line
----
Below the ruled line
```

will be displayed as follows:

Above the ruled line

Below the ruled line

4.4.7 Adding the author's name

When we are creating content and contributing to the information in the wiki, then we don't need to add our own name anywhere since it is a collaborative effort. However, if we are using the wiki to carry out a discussion about a topic, then it is useful to know who made a given comment. This is particularly relevant for the talk pages that are used for general discussion about the content of a page.

When we are editing a page, we can get the system to automatically add our names at a given location by simply using a series of three tilde characters in a row `~~~`. The system will automatically insert a link to our user name at this point. If we also want the time and date (very useful for discussions), then four tilde characters in a row should be used `~~~~`. For example, the wiki markup:

Page Source

```
Your formula is incorrect. You have forgotten
to take account of the GST.  ~~~

Ooops.  Sorry about that.
: ~~~~
```

will be displayed as:

> You forgot to add the GST in your formula. *Alux*001
>
> Ooops. Sorry about that.
> *Alux*001 11:33, 8 Sep 2005 (NZST)

Note that we added the markup `:` to indent the author in the second example.

4.4.8 Links

There are different kinds of links that can be created in a wiki. The following table summarises the different links available.

Wiki Markup	Meaning
`[[Link]]`	Internal link
`[[Link|Label]]`	Internal link with label
`URL`	External link
`[URL Label]`	External link with label

Table 4.3: Links supported by MediaWiki

Internal links

The simplest kind of link is to another page within the wiki. To do this, simply put the name of the page inside nested square brackets `[[name goes here]]`. For example, the wiki markup:

Page Source

```
1 For more information see [[Bees]]
```

will be displayed as follows:

For more information see *Bees*

This has created a link to the page titled “Bees” within the wiki.

Labels for internal links

By default, the link that we see on the page will be the same as the name of the page that we are linking to. However, if we label the link, then the the viewer will see the label instead of the actual name of the page. The general format for this is:

```
[[Page Name|Link Label]]
```

When this link is displayed, the text will show the “Link Label” as the link on the page. When the link is clicked, the wiki will display the page called “Page Name”. For example, the wiki markup:

Page Source

```
1 For more information about links click [[Help|here]]
```

will be displayed as follows:

For more information about links click *here*

External links

The easiest way to create an external link is to simply include the URL of the link in the normal text. The wiki will recognise that the text contains a URL and will automatically create a link for that address. For example, the wiki markup:

Page Source

```
1 The link to all the course pages can be found on
2 the main computer science department website.
3 The address is http://www.cs.auckland.ac.nz/courses
```

will be displayed as follows:

The link to all the course pages can be found on the main computer science department website. The address is *http://www.cs.auckland.ac.nz/courses*

Labels for external links

If we want to include a label for an external link, then we need to enclose the link in square brackets. After the link, we leave a single blank space and then include the label. The label will be displayed on the page, and if clicked, it will link to the address given by the URL. The formal syntax for this is as follows:

```
[URL Label]
```

For example, the following wiki markup:

Page Source

```
1 All of the courses offered this year in Computer
2 Science have a  web page.  A link to each web page
3 can be found on the main
4 [http://www.cs.auckland.ac.nz Computer Science department]
5 website.
```

will be displayed as follows:

All of the courses offered this year in Computer Science have a web page. A link to each web page can be found on the main *Computer Science department* website.

4.4.9 Character formatting

MediaWiki allows us to format individual characters or words. The following table summarizes the character formatting supported by MediaWiki.

Wiki Markup	Meaning
`''Text''`	Emphasis (italic)
`''' Text '''`	Strong (bold)
`''''' Text '''''`	Very strong (bold and italic)

Table 4.4: Character formatting supported by MediaWiki

The most commonly used markup is described below.

Emphasize

In order to emphasize some text, we can use two single quote marks around the text. The browser will normally display the emphasized text in an italic font. The formal syntax for this is:

```
''Some text''
```

For example, the wiki markup:

Page Source

```
1 This topic is ''extremely'' important
```

will be displayed as follows:

This topic is *extremely* important

Strong

In order to make some text stand out in a strong way, we can use three single quote marks around the text. The browser will normally display the strong text in a bold font. The formal syntax for this is:

```
'''Some text'''
```

For example, the wiki markup:

Page Source

```
1 This topic is '''extremely''' important
```

will be displayed as follows:

This topic is **extremely** important

Very strong

In order to make some text stand out in a very strong way, we can use five single quote marks around the text. The browser will normally display the very strong text in both italic and bold font. The formal syntax for this is:

```
'''''Some text'''''
```

For example, the wiki markup:

Page Source

```
1 This topic is '''''extremely''''' important
```

will be displayed as follows:

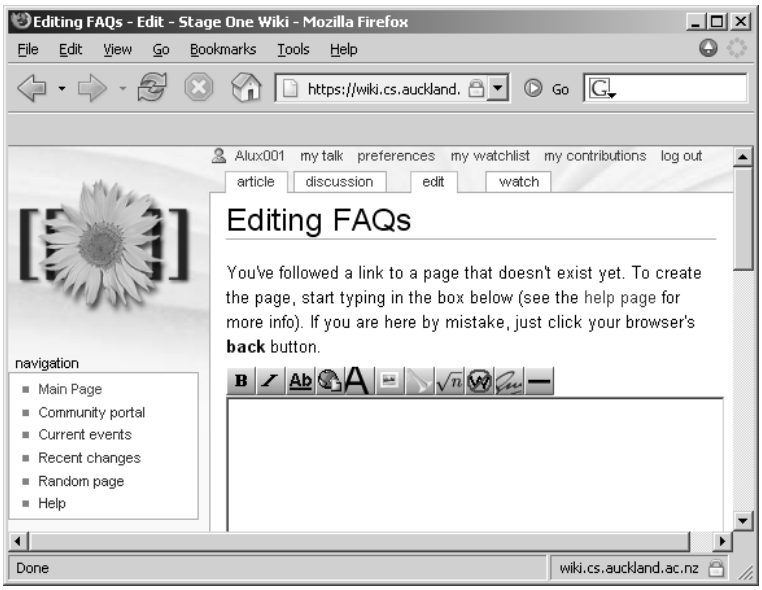
This topic is ***extremely*** important

4.5 Creating a new page

Creating a new page in a wiki is a simple process.

4.5.1 Following a link

A new page is created when we click on a link that does not have a destination page. In this case, the wiki will tell us that the destination of that link does not exist and we will be given the option to create the content of that page.



The links that do not have a destination page are coloured red in MediaWiki.

4.5.2 Creating a new link

If we want to create a completely new page on a given topic (for example, a page containing "Frequently Asked Questions about HTML"), then we need to first create a link to that page.

We should browse the wiki and find the appropriate place to add a link. For example, if we wanted to add a page called "Frequently Asked Questions about HTML" then we might decide to create a link on a page called "HTML" that leads to our new page.

Once we have found the appropriate page, we simply edit that page and add a link to our proposed new page. In the example here, the link would be `[[Frequently Asked Questions about HTML]]`. Once this page has been saved, our wiki will have a new link that leads to a non-existent page. When someone clicks on the link, then they will be given an option to create the content of that page.

4.6 References

- http://meta.wikimedia.org/wiki/MediaWiki_User's_Guide
- http://en.wikipedia.org/wiki/Wiki

CHAPTER 5

Hypertext Markup Language (HTML5)

5.0.1 Versions of HTML and XHTML

The first version of the Hypertext Markup Language was written by Tim Berners-Lee in 1993 when he was developing the system that grew into the WWW. This was never established as an official standard and it lacked the ability to display images. As the web become available to the general public in 1995, HTML 2.0 was released as an official standard. It continued to be used until 1997, when it was replaced by HTML 3.2 and later the same year, HTML 4.0. Some minor modifications were made, and HTML 4.01 was released in 1999. This version has been widely used and it defines the standard that most web sites adhere to.

In 2000, the W3 Consortium released a newer, cleaner version of HTML called XHTML 1.0. This language is almost identical to HTML 4.01, with minor changes to ensure consistency and ease of use. The W3C defines XHTML as the latest version of HTML, and states that XHTML is intended to gradually replace HTML.

The latest official version is XHTML 1.1 which is designed to support the use of mobile devices (such as mobile phones and personal organisers) to access web pages. Internet Explorer causes problems viewing XHTML 1.1 pages, so most people use XHTML 1.0 instead. The W3 Consortium are currently working on XHTML5.

On 14 February 2011, the W3C extended the charter of its HTML Working Group with clear milestones for HTML5. In May 2011, the working group advanced HTML5 to “Last Call”, an invitation to communities inside and outside W3C to confirm the technical soundness of the specification. The W3C has developed a comprehensive test suite to achieve broad interoperability for the full specification in 2014.

5.0.2 Document Type Definition

Since there are so many different versions of HTML and XHTML, it is important to tell the browser what version of the language is being used. The format for representing that information is known as the DTD (Document Type Definition). The DTD specifies the type of the language that is being used for that document.

If we are using XHTML 1.0 Strict, we must include that information first in any file that will be displayed using a browser. The DTD is usually copied and pasted from a reference page. For example, in XHTML1.0:

```
<!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

In HTML5 the DTD specification is:

```
<!DOCTYPE html>
```

5.0.3 Encoding standards

In addition to the DTD, we also need to tell the browser what encoding system was used to store the page. This is important for compatibility across different platforms and different languages. We will use the following line of code and it is stored in the `<head>` tags:

```
<meta charset="UTF-8">
```

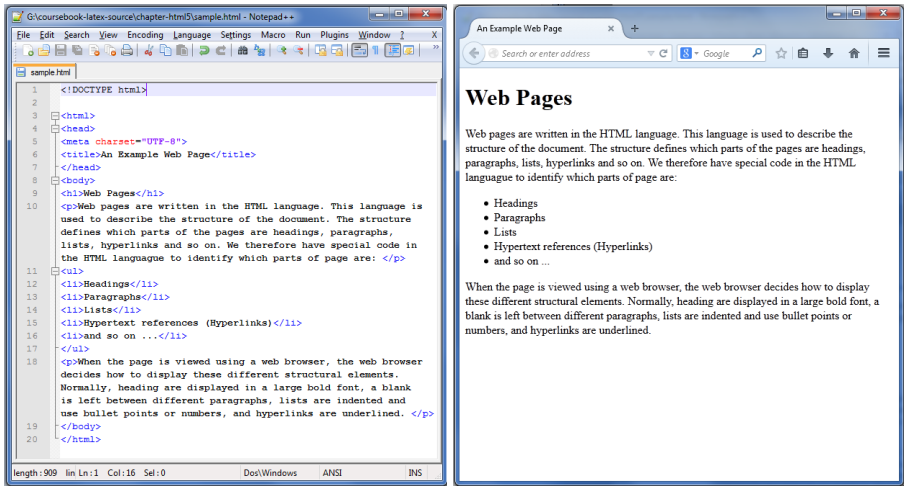
So at the start of every HTML5 page that you write, you should copy and paste the text:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
```

5.1 Hypertext Markup Language (HTML)

Hypertext Markup Language (HTML) is used to describe the structure and formatting of a document which forms part of the World-Wide Web. In other words, *web pages* are written in HTML. Although most web pages include text of different sizes, pictures and coloured backgrounds, the HTML code which tells the browser how to display the information is written in plain ASCII text.

The file containing the HTML is known as the *source file*. Since the source file contains only plain text, a web page can be written using any plain text editor such as Notepad++. When a web browser is used to view the page, it will understand the HTML and display the content of the page accordingly.



The HTML source file appears on the left. The same page viewed by a web browser appears on the right

5.2 Tags

HTML consists of a series of "codes" known as *tags*. These tags contain the information that a web browser needs in order to display the page correctly. All tags are enclosed within the < and > characters. For example, `<i>` is an HTML tag which is used to make text italic.

Tags normally come in pairs, such as `<i>` and `</i>`. The first tag `<i>` tells the browser to start making the text italic. The second tag (usually called the closing tag) tells the browser to stop making the text italic. We normally think of the start and end tags as *containing* the text which the tag applies to. For example, if we wanted to make the word "distinct" appear in italics, we would use the `<i>` tag in the following way:

```
Two or more things are <i>distinct</i> if no two
of them are the same thing.
```

In general, tags have the form:

`<tag>` ... `</tag>`

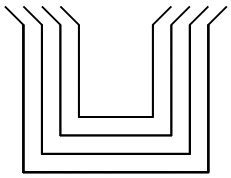
The text in between the two tags (represented by "...") is the text which is affected by that tag. In HTML, tags must always be in lower case.

5.2.1 Nested tags

When multiple tags are used, then it is important they they are nested correctly. It can help to think of the tags as being containers. The text which appears between an opening and closing pair of tags is *contained* by those tags. For example, in the following sentence, the tag `<i>` can be said to contain the text "important".

```
It is <i>important</i> that tags are
nested correctly.
```

Real containers can be placed inside one another.



Containers can hold other containers

Tags that "contain" text are similar. When we want to apply multiple tags to a piece of text, we have to ensure that each tag is fully contained by another one. For example, if we wanted to make the word "important" both bold and italic, then we could use the HTML code:

```
It is <b><i>important</i></b> that tags are
nested correctly.
```

or we could use:

```
It is <i><b>important</b></i> that tags are
nested correctly.
```

but it would be incorrect to overlap them as follows:

```
It is <i><b>important</i></b> that tags are
nested correctly.
```

When we use multiple tags, we must ensure that they follow the general form

`<tag1> <tag2> ... </tag2> </tag1>`

In this case, we would say that `<tag2>` is nested within `<tag1>`.

5.2.2 Attributes of tags

Some tags require additional information to be used. For example, a tag used to display an image would need to know the file name of the image. The format for tags of this

sort are as follows:

```
<tag property="value"> ... </tag>
```

For example, the following anchor tag `<a>` uses the attribute `href`:

```
<a href="http://www.w3.org">The W3 Consortium</a>
```

Sometimes tags use more than one attribute. In this case, the attributes are simply separated by a space as follows:

```
<tag property1="value1" property2="value2"> ... </tag>
```

For example, the following image tag `<img>` uses two attributes:

```
<img src="smile.jpg" alt="A smiley face"></img>
```

The use of attributes will become more common in later sections of this document.

5.3 Essential HTML tags

There are four essential tags which define the structure of an HTML document. All HTML documents should include these tags. They are:

- `<html>`
- `<head>`
- `<title>`
- `<body>`

Each of these tags will be described in more detail below.

5.3.1 `<html>`

The `<html>` tag should surround all other HTML text in the document. This tag is used to define the start and end of the HTML document. For example:

Page Source

```
1 <!DOCTYPE html>
2
3 <html>
4
5 ... the rest of the code goes here
6
7 </html>
```

5.3.2 <head>

The head of a document contains information which the browser needs in order to manage the display of the page. The information within the head of an HTML document is not intended to be viewed by a human user. None of the material in the head will be displayed on the page itself, rather this is information used only by the browser. For example:

Page Source

```
<!DOCTYPE html>

<html>
<head>
<meta charset="UTF-8">

... this information is used by the browser

</head>

... other HTML tags appear here

</html>
```

5.3.3 <title>

The title of the document is the only essential tag which should appear in the head. The title defines the name of the document which is being displayed. This title typically appears in the title bar of the web browser. The title is used by the browser to help the user to navigate through the WWW. It is typically used when the page is bookmarked or in the history list of the web browser (i.e. when you want to go back to a page that you have previously visited). The title will not appear in the main page which is displayed. For example:

Page Source

```
<!DOCTYPE html>

<html>
<head>
<meta charset="UTF-8">
<title> ... the title of the document ... </title>
</head>

... other HTML tags appear here

</html>
```

5.3.4 <body>

The body of a document contains the actual content of the page. Any material which appears on a web page will be defined inside the body of the corresponding HTML document. For example:

Page Source

```
<!DOCTYPE html>

<html>
<head>
<meta charset="UTF-8">
<title> ... the title of the document ... </title>
</head>
<body>

... the body contains the content that is
    displayed on the page.

</body>
</html>
```

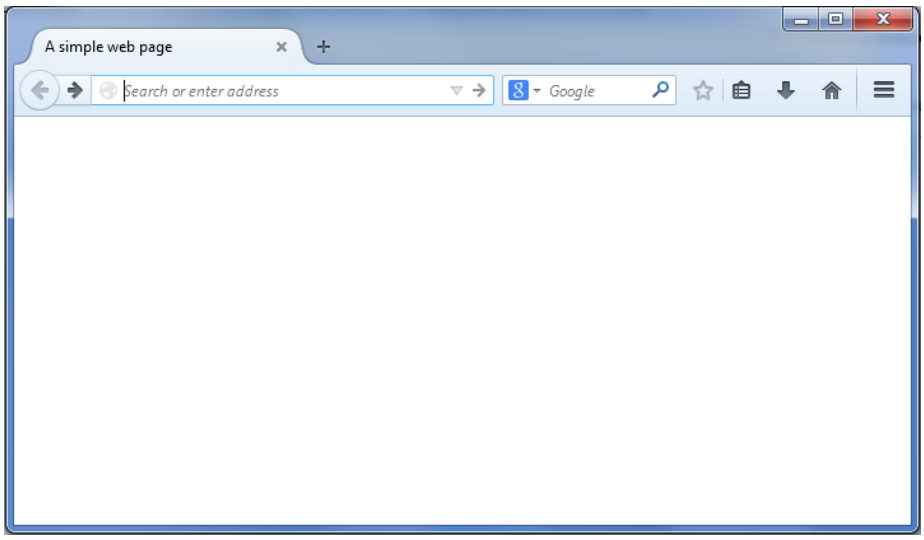
5.3.5 A simple example

A complete HTML document is shown below:

Page Source

```
<!DOCTYPE html>

<html>
<head>
<meta charset="UTF-8">
<title>A simple web page</title>
</head>
<body>


</body>
</html>
```

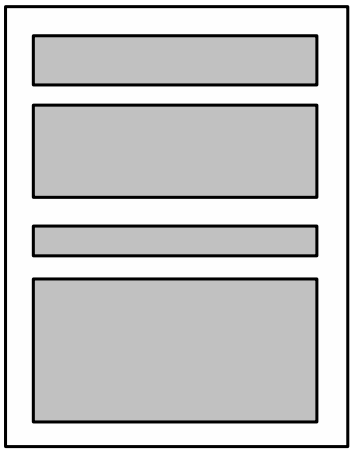
When the page is viewed using a web browser, it will display an empty page. We need to know more about the tags used to format the content before we can include information in the body of the page. The empty page will look like:



Note that the title appears in the title bar or tab of the window. The only information which is displayed in the main area of the web browser is the content which appears in the body of the HTML document (in this case, nothing at all).

5.4 Block-level tags

When we consider the layout of a page, we think of it as being broken up into different blocks. A blank line normally appears between each block.



A page is divided up into blocks

We define the structure of our documents using block-level tags and inline tags. The block-level tags are used to define the different blocks that make up our page. The inline tags are normally used within blocks. The most commonly used block-level tags are described below.

5.4.1 <h1> to <h6>

There are six different levels of heading supported by HTML. The most important heading is defined as “level one”. The least important heading is defined as “level-six”. The tags used for headings are listed below:

<h1> Level-one heading. The most important heading.

<h2> Level-two heading

<h3> Level-three heading

<h4> Level-four heading

<h5> Level-five heading

<h6> Level-six heading. The least important heading.

Each document should have exactly one heading of level-one importance. You should not choose the heading based on the appearance of the text (since the appearance can be changed later), but rather consider the level of importance of the heading and use the appropriate level of heading.

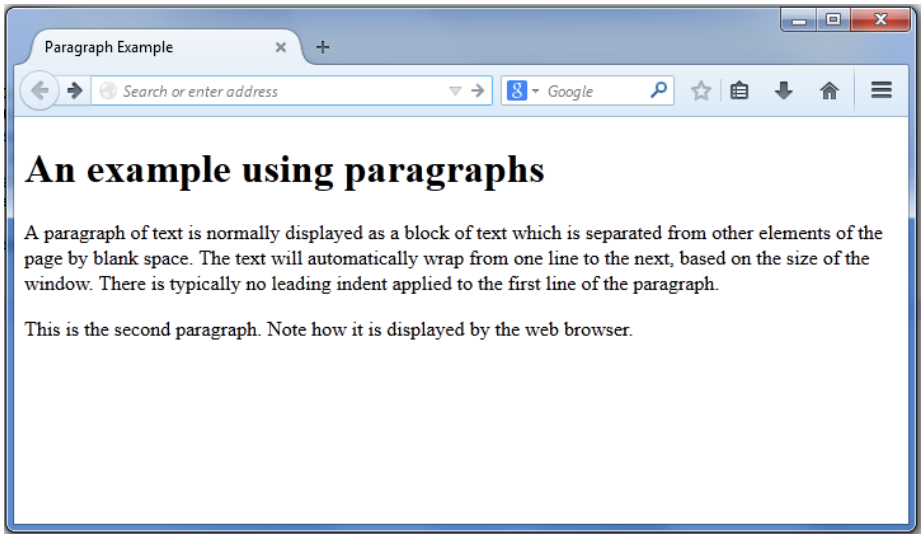
5.4.2 <p>

To define a paragraph in HTML, we use the <p> tag. Since a paragraph is a block-level tag, paragraphs are normally separated with a blank line. For example, the page:

Page Source

```
<!DOCTYPE html>

<html>
<head>
<meta charset="UTF-8">
<title>Paragraph Example</title>
</head>
<body>
<h1>An example using paragraphs</h1>
<p>A paragraph of text is normally displayed as a
block of text which is separated from other
elements of the page by blank space.  The text will
automatically wrap from one line to the next, based
on the size of the window.  There is typically no
leading indent applied to the first line of the
paragraph.</p>
<p>This is the second paragraph.  Note how it is
displayed by the web browser.</p>
</body>
</html>
```

will be displayed as follows:



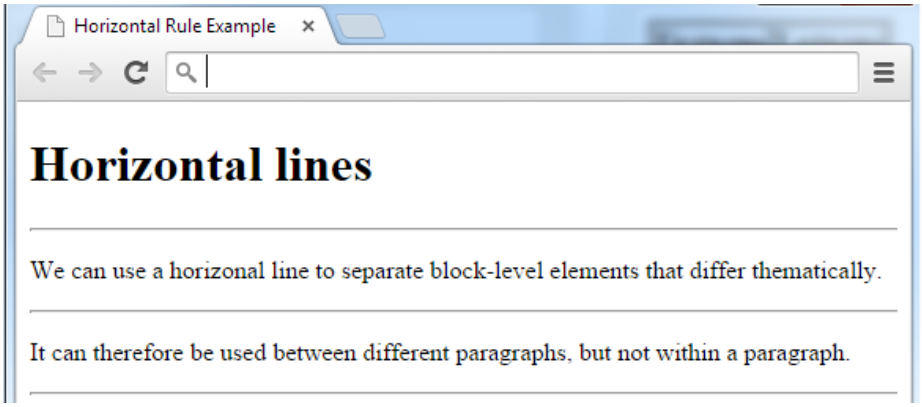
5.4.3 <hr>

In HTML5, the <hr> tag defines a thematic break in content. It usually appears as a horizontally ruled line when a page is displayed in a browser. Since this tag does not format any content it does not require a closing tag. The following HTML source code shows how the <hr> tag is used.

Page Source

```
<!DOCTYPE html >

<html>
<head>
<meta charset="UTF-8">
<title>Horizontal Rule Example</title>
</head>
<body>
<h1>Horizontal lines</h1>
<hr>
<p>
We can use a horizonal line to separate
block-level elements that differ thematically.
</p>
<hr>
<p>It can therefore be used between different
paragraphs, but not within a paragraph.
</p>
<hr>

</body>
</html>
```

The code shown previously will be displayed in a web browser as:



5.4.4 <pre>

The <pre> tag is used to define a block of text as being pre-formatted. This means that white space used in the original source code is kept when the block of text is displayed by the browser. The font used to display the pre-formatted text is a fixed-width font such as courier.

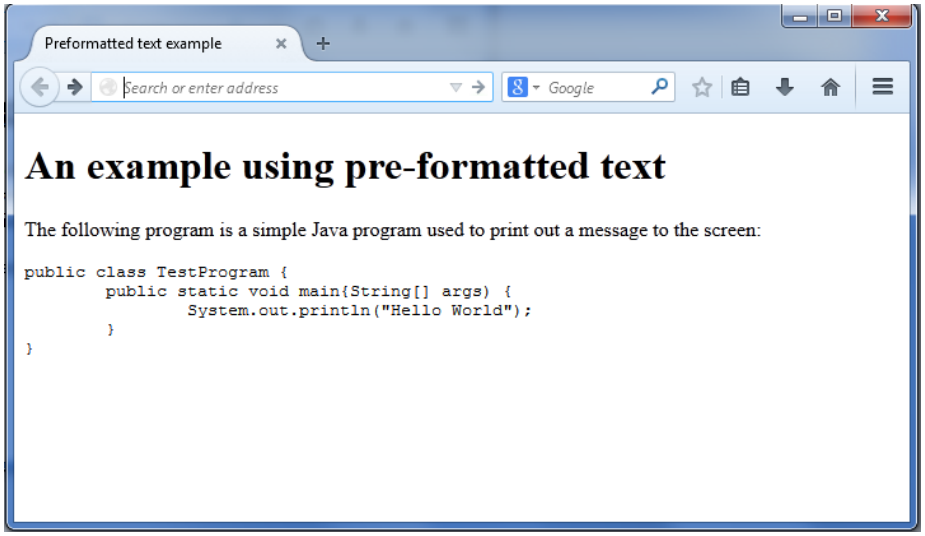
This is normally used when web pages are used to show computer code for programs. For example, the HTML source file below uses the <pre> tag to surround a block of text containing a Java computer program.

Page Source

```
<!DOCTYPE html>

<html>
<head>
<meta charset="UTF-8">
<title>Preformatted text example</title>
</head>
<body>
<h1>An example using pre-formatted text</h1>
<p>The following program is a simple Java program
used to print out a message to the screen:</p>
<pre>
public class TestProgram {
    public static void main{String[] args) {
        System.out.println("Hello World");
    }
}
</pre>
</body>
</html>
```

The screenshot on the next page shows what the file would look like when viewed using

a web browser.



5.4.5 Tables

The `<table>` tag is used to format information into rows and columns. This tag is very important. It is widely used in the design of web pages, and it is a powerful tool for laying out documents.

`<table>`

The `<table>` tag should surround all the information about the table.

```
<table>
... table defined here ...
</table>
```

`<tr>`

A table consists of a series of rows. Each row is defined using the `<tr>` tag. A table must have at least one row. There is no maximum number of rows.

```
<table>

<tr>... first row defined here ... </tr>
<tr>... second row defined here ... </tr>
<tr>... third row defined here ... </tr>

</table>
```

<td>

A single row in a table consists of a series of different cells. Each cell in the row is defined using the table data tag `<td>`.

```
<table>

<tr>
<td>First cell</td><td>Second cell</td>
</tr>

<tr>
<td>Another cell</td><td>And another cell</td>
</tr>

<tr>
<td>Still another cell</td><td>Yet another cell</td>
</tr>

</table>
```

The width of each cell in the table is automatically adjusted to ensure that the borders of the cells align with each other. Cells which contain more information will generally be wider.

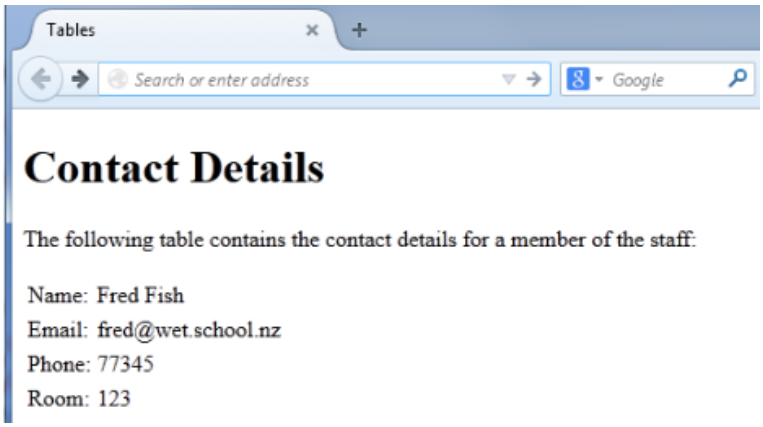
An example of a table

The following HTML source code shows how the `<table>` tag can be used to present information in a table.

Page Source

```
1  <!DOCTYPE html>
2
3  <html>
4  <head>
5  <meta charset="UTF-8">
6  <title>Tables</title>
7  </head>
8  <body>
9  <h1>Contact Details</h1>
10 <p>The following table contains the contact details
11 for a member of the staff:</p>
12 <table>
13 <tr><td>Name:</td><td>Fred Fish</td></tr>
14 <tr><td>Email:</td><td>fred@wet.school.nz</td></tr>
15 <tr><td>Phone:</td><td>77345</td></tr>
16 <tr><td>Room:</td><td>123</td></tr>
17 </table>
```

```
18 </body>
19 </html>
```



5.4.6 Lists

A list consists of a series of items. There are three kinds of lists that you can use:

- an ordered list,
- an unordered list,
- and a definition list.

Each of the different lists in HTML are discussed below.

The `<ol>` tag is used to define an ordered list. Each item in the list is numbered automatically by the web browser. The tag `<li>` is used to define each item in the list. The general structure is as follows:

```
<ol>
<li>Apple</li>
<li>Orange</li>
<li>Pear</li>
</ol>
```


The `<ul>` tag is used to define an unordered list. Each item in the list begins with a bullet point. The tag `<li>` is used to define each item in the list. The general structure is very similar to the `<ol>` described previously.

```
<ul>
<li>Apple</li>
<li>Orange</li>
<li>Pear</li>
</ul>
```

<dl>

The <dl> tag is used to define a definition list. This is used when there are a series of terms that need to be defined. Each entry in the definition list consists of a definition term and a definition description. The <dt> tag is used for the definition term. The <dd> tag is used for the definition description. The general structure is as follows:

```
<dl>
<dt>Apple</dt>
<dd>An innovative computer company</dd>
<dt>Orange</dt>
<dd>A colour which is a mixture of red and yellow</dd>
<dt>Pear</dt>
<dd>A fruit</dd>
</dl>
```

Example of lists

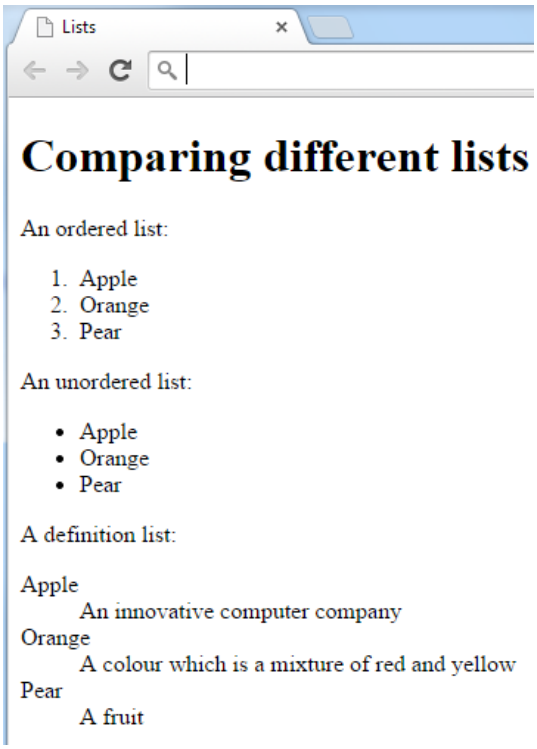
The following HTML source code includes all three types of list.

Page Source

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="UTF-8">
5  <title>Lists</title>
6  </head>
7
8  <body>
9  <h1>Comparing different lists</h1>
10 <p>An ordered list:</p>
11 <ol>
12 <li>Apple</li>
13 <li>Orange</li>
14 <li>Pear</li>
15 </ol>
16
17 <p>An unordered list:</p>
18 <ul>
```

```
<li>Apple</li>
<li>Orange</li>
<li>Pear</li>
</ul>

<p>A definition list:</p>
<dl>
<dt>Apple</dt>
<dd>An innovative computer company</dd>
<dt>Orange</dt>
<dd>A colour which is a mixture of red and yellow</dd>
<dt>Pear</dt>
<dd>A fruit</dd>
</dl>
</body>
</html>
```

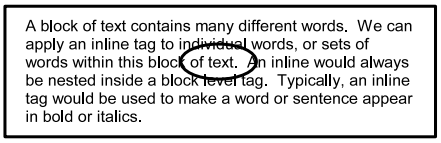
The following screenshot shows the page when viewed using a web browser.



5.5 Inline tags

An inline tag is applied to something which exists inside a block. For example, if you had a paragraph of text, you could use an inline tag to apply italics to a word within the

paragraph. The tag would be applied without breaking the line of text, so it is known as an inline tag.



The most commonly used inline tags are described below.

5.5.1

To end the current line of text, we can use the `<br>` tag. This forces a line break at the location of the tag (it causes the same effect as if the Enter key was pressed). Since this tag does not apply to any existing text on the page it does not require a closing tag. The following HTML source code shows how the `<br>` tag is used:

Page Source

```
<!DOCTYPE html>

<html>
<head>
<meta charset="UTF-8">
<title>Break Example</title>
</head>
<body>
<h1>An example using paragraphs and breaks</h1>
<p>
Blank lines in the

source code are replaced

by a simple space.  If we want to have a break<br>
then we use the break tag.</p>
<p>
Author: Andrew Luxton-Reilly<br>
Date: 01/01/06
</p>
</body>
</html>
```

will be displayed by a web browser as:

5.5.2 <img>

The <img> tag is used to include an inline image in the page. It uses a series of attributes which tell the browser which image to display, and how it should be displayed. The image referred to in the tag will be displayed on the web page at the location that the tag appears.

The <img> tag uses the following attributes:

src The source of the image. This attribute is used to specify where the image is located so that the web browser can load the image from that location. A normal URL should be used here. Typically, this is a relative reference which refers to an image in the same directory as the web page. Every <img> tag must include the src attribute. Note that most web browsers are only able to display pictures which are stored in .jpg, .gif or .png format.

alt Alternative text which is displayed if the browser is unable to display the image. This text should describe the image so that the viewer of the page knows what content they are missing out on if the browser cannot display the image (or if the user chooses not to load the images). All image tags should include the alt attribute.

height The height of the image. If the height specified is not the same as the original image then the browser will scale the image to the height required. Note that the browser software is not designed to scale images, so the quality of the modified image may be worse than if it was scaled using image processing software.

Using a specified height allows the browser to set aside enough room for the picture and continue to render the remainder of the page. If a height is not specified, then the browser has to wait until it has downloaded the image before it can start to draw anything on the page. Using the height attribute will often mean that pages are displayed quicker when they are downloaded. If the height is not specified, but the width has been specified then the height will automatically be adjusted to maintain the same ratio of height to width as the original image.

The use of this attribute is recommended, although not required.

width The width of the image. The comments about the height attribute above apply here with respect to the width.

For example the source code:

```
<p>The following picture shows a camel:<br>
<img src="camel.jpg" alt="A camel">
</p>
```

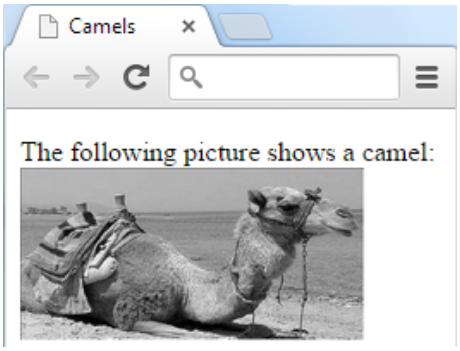
would produce the following:

Note that there is no closing tag `</img>`. This is because the tag is not making a change to any text that appears on the page, rather the picture that will be displayed by the tag will be obtained from a file located on disk.

If we use the width and height attributes, we can scale the image and make it smaller. The following source code:

```
<p>The following picture shows a camel:<br />

<img src="camel.jpg" alt="A camel" width="200"
 height="100">
 </p>
```

would produce the following picture:



A complete HTML page is provided on the next page.

Page Source

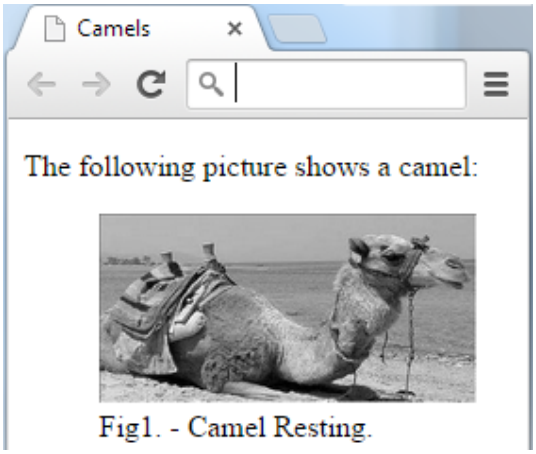
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Camels</title>
</head>
<body>
<h1>Ships of the Desert</h1>

<p>
The camel has a reputation for being a foul-tempered
beast, but in fact camels are good-natured animals.
They are intelligent, friendly animals which make
good companions for desert travel.
</p>
<p>
<img src="camel.jpg" alt="A camel" width="200"
height="100">
</p>
<p>
A camel can last five to seven days without water. This
is due to an extremely efficient biology. The camel does
not store water in its hump.  Rather, the hump is a fatty
deposit used to store energy.  The fat reserves are used
when food is scarce.
</p>

</body>
</html>
```

The HTML document above would be displayed by a web browser as:

In books and newspapers, it is common to have captions with images. The purpose of a caption is to add a visual explanation to an image. With HTML5, images and captions can be grouped together in `<figure>` elements:

```
<p>The following picture shows a camel:<br />
<figure>
    <img src="camel.jpg" alt="A camel" width="200"
     height="100">
    <figcaption>Fig1. - Camel Resting.</figcaption>
</figure>
 </p>
```

The HTML document above would be displayed by a web browser as:



5.5.3 `<a>`

Perhaps the most important tag of the HTML language is the "anchor" tag `<a>`. This tag is used to create the hyperlinks that connect different pages to each other. The `<a>` tag uses the following attribute:

`href` This attribute is used to create a hypertext reference (in other words a clickable link to another location in the WWW). The `href` attribute specifies the destination of the hyperlink. The destination may be a relative or absolute URL.

The destination of the link

```
<a href="http://www.cs.auckland.ac.nz/people">staff members</a>
```

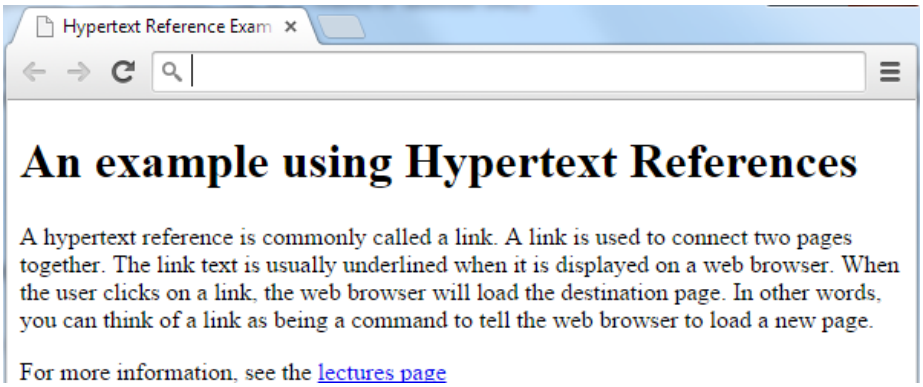
The text that appears underlined on the web page

The following example shows some HTML source code that contains a link to another document.

Page Source

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Hypertext Reference Example</title>
</head>
<body>
<h1>An example using Hypertext References</h1>

<p>
A hypertext reference is commonly called a link.  A
link is used to connect two pages together.  The link
text is usually underlined when it is displayed on a
web browser.  When the user clicks on a link, the web
browser will load the destination page.  In other
words, you can think of a link as being a command
to tell the web browser to load a new page.
</p>

<p>
For more information, see the
<a href="http://www.cs.auckland.ac.nz/lectures/">
lectures page</a>
</p>

</body>
</html>
```

When this page is viewed using a web browser, then it will appear as follows:



5.6 Uniform Resource Locator

A uniform resource locator (URL) is a standard address that specifies the location of a resource on the Internet. The URL is formed from different parts.

5.6.1 Protocol

The first part of a URL is the protocol. The protocol specifies how the data will be transferred. For example, if the resource is a file, then it will be transferred using the File Transfer Protocol (FTP). If the resource is a web page, then it will be transferred using the HyperText Transfer Protocol (HTTP).

5.6.2 Host Name

The second part of a URL specifies which host computer on the Internet is used to store the resource. Typically, this host name is a Domain Name such as `www.cs.auckland.ac.nz`.

5.6.3 Path

The third part of a URL is the path. The path specifies where on the host computer the file is located. In other words, the path specifies the directory where the resource is stored.

5.6.4 Resource Name

The last part of a URL is the name of the actual resource that we are looking for. This is normally a file name.

5.6.5 Examples

A typical URL is:

```
http://www.cs.auckland.ac.nz/compsci111s1c/lectures/
index.html
```

This URL can be broken up into its constituent parts as follows:

Description	Value used
protocol	`http`
host	`www.cs.auckland.ac.nz`
path	`/compsci101s1c/lectures/`
resource	`index.html`

5.7 Comments

It can be helpful to include comments in your HTML code. These comments are designed to help other people understand the code and are not displayed by the web browser. A special tag is used to include comments in your code.

```
<!--
Comments go here
-->
```

This HTML tag does not follow the normal rules of syntax. There is no start and end tag, and there are no attributes. Instead, the tag itself begins with the symbols `<!--` and the tag ends with the symbols `-->`. Any other text that is used between these symbols will be ignored by the browser (it will not display the text that is part of a comment).

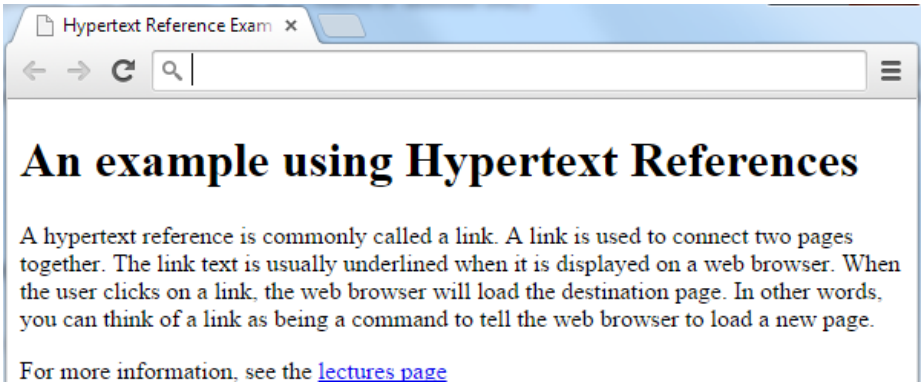
For example, the following HTML code:

Page Source

```
1  <!DOCTYPE html>
2  <!--
3      Created Date: 01/04/06
4      Modified Date: 13/10/14
5  -->
6  <html>
7  <head>
8  <meta charset="UTF-8">
9
10 <!-- The title is used when we bookmark the page -->
11 <title>Hypertext Reference Example</title>
12 </head>
13 <body>
14 <h1>An example using Hypertext References</h1>
15
16 <p>
17 A hypertext reference is commonly called a link.  A
18 link is used to connect two pages together.  The link
19 text is usually underlined when it is displayed on a
20 web browser.  When the user clicks on a link, the web
21 browser will load the destination page.  In other
22 words, you can think of a link as being a command
23 to tell the web browser to load a new page.
24 </p>
25
26 <!-- Note:  We could add other references here -->
27 <p>
28 For more information, see the
29 <a href="http://www.cs.auckland.ac.nz/lectures/">
30 lectures page
31 </a>
32 </p>
33
```

```
34 </body>
35 </html>
```

will produce the same page as the example in the previous section as shown below:



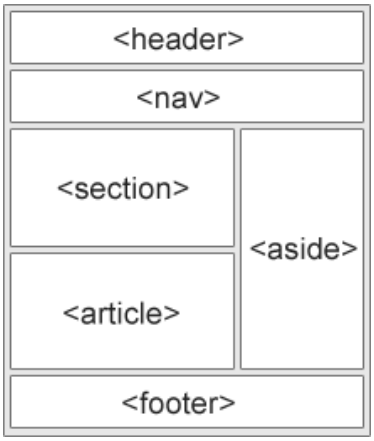
5.8 HTML5 Semantic Elements

Semantics is the study of meaning. Semantic elements are elements with a meaning. A semantic element clearly describes its meaning to both the browser and the developer.

Examples of non-semantic elements: `<div>` and `<span>`. These elements tell nothing about their content. Examples of semantic elements are `<form>`, `<table>`, and `<img>`. These elements clearly define their content.

Many web sites contain HTML code like: `<div id="nav">`, `<div class="header">`, `<div id="footer">` to indicate navigation, header, and footer.

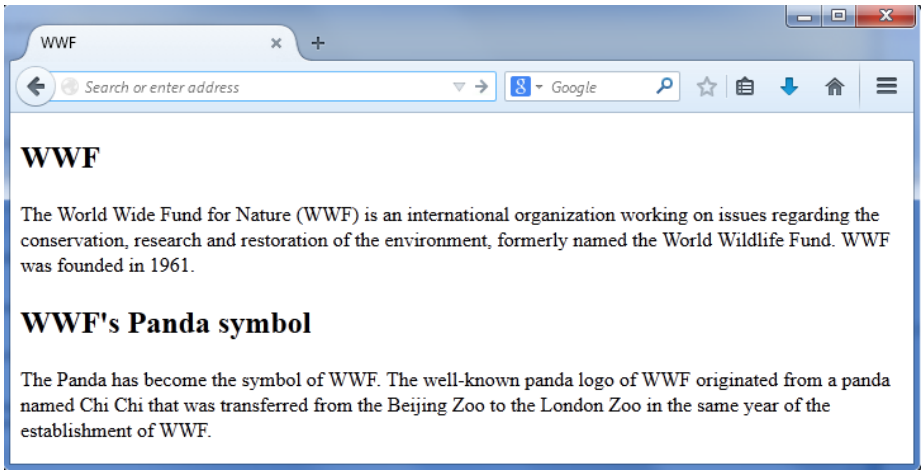
HTML5 offers semantic elements to define different parts of a web page.



For example, the `<section>` element defines a section in a document. A Web site's home page could be split into sections for introduction, content, and contact information.

Page Source

```
<!DOCTYPE html>
<html>
<head>
<title> WWF </title>
</head>
<body>

<section>
<h1>WWF</h1>
<p>
The World Wide Fund for Nature (WWF) is an international
 organization working on issues regarding the conservation,
research and restoration of the environment, formerly
named the World Wildlife Fund. WWF was founded in 1961.
</p>
</section>

<section>
<h1>WWF's Panda symbol</h1>
<p>
The Panda has become the symbol of WWF. The well-known
panda logo of WWF originated from a panda named Chi Chi
that was transferred from the Beijing Zoo to the London
Zoo in the same year of the establishment of WWF.
</p>
</section>

</body>
</html>
```



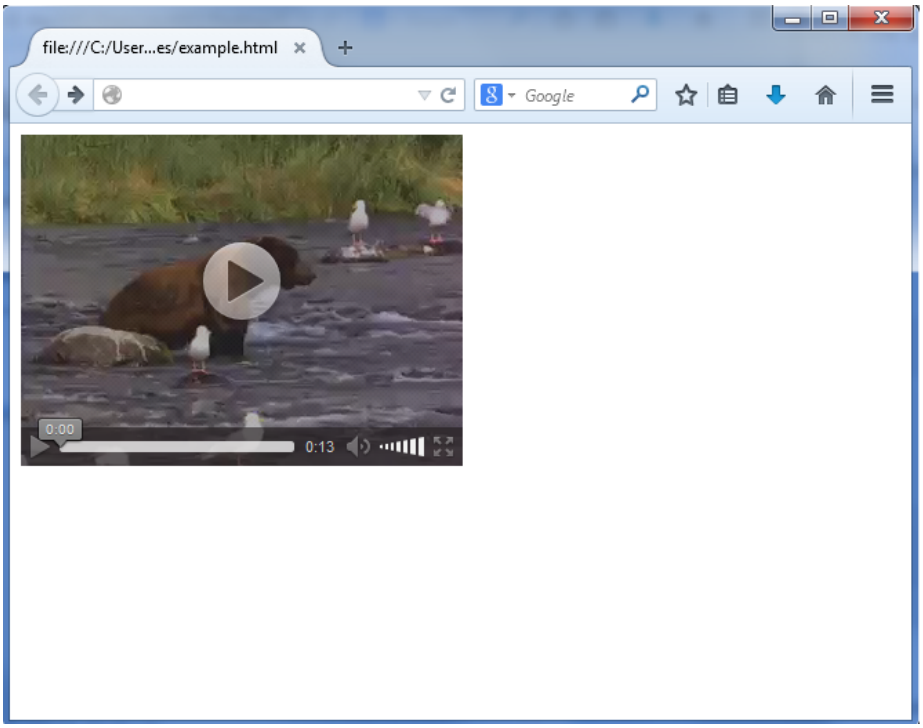
5.9 Videos in HTML

Before HTML5, there was no standard for showing videos on a web page and videos could only be played with a plug-in (like flash). The HTML5 `<video>` element specifies a standard way to embed a video in a web page.

To show a video in HTML, use the `<video>` element:

Page Source

```
<!DOCTYPE html>
<html>
<body>

<video width="320" height="240" controls>
    <source src="movie.mp4" type="video/mp4">
    <source src="movie.ogg" type="video/ogg">
     Your browser does not support the video tag.
</video>

</body>
</html>
```



To start a video automatically use the autoplay attribute:

```
<video width="320" height="240" autoplay>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
```

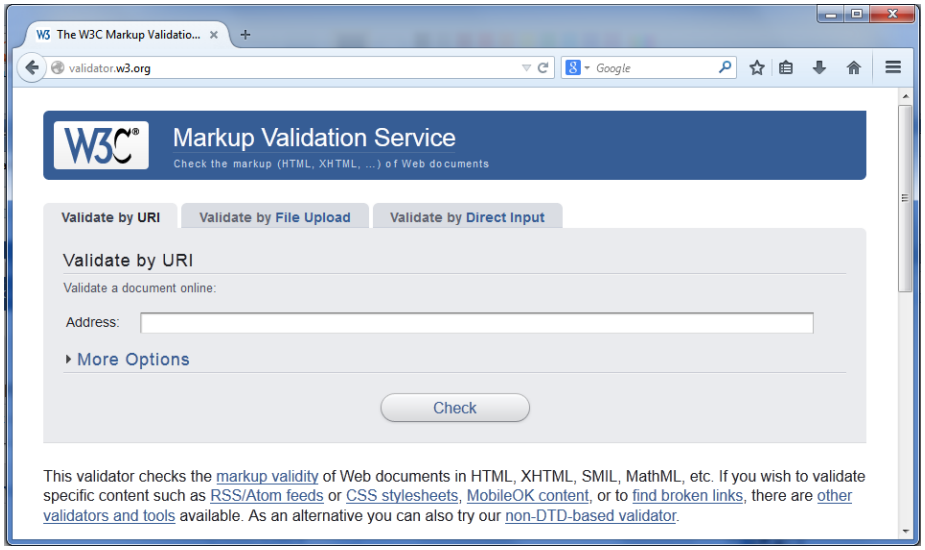
```
    Your browser does not support  video tag.
</video>
```

5.10 Validating your pages

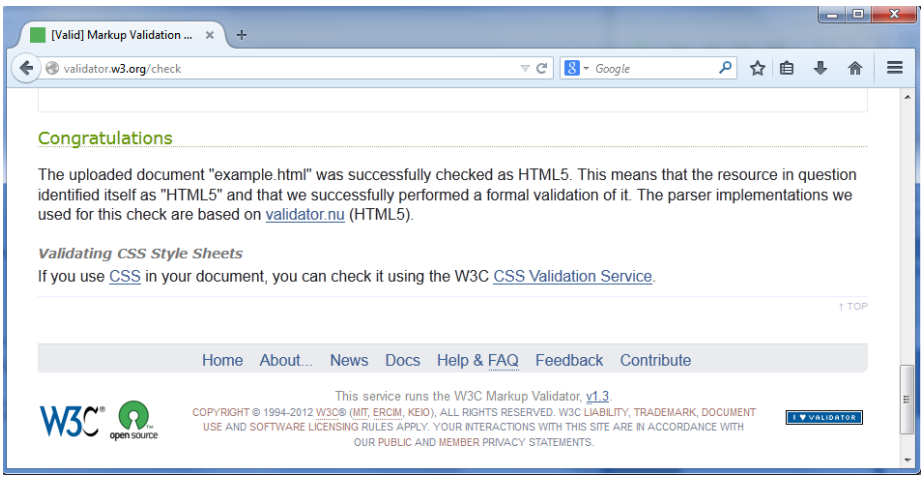
The W3C provides a free service that can be used to check your HTML or XHMTL code to see if it meets the official standard (in other words, to see if it is correct). The URL of the validation service is:

http://validator.w3.org/

You can check pages using three different methods. Give the URL of the page, upload the file containing the page, or copy and paste the page source into the form provided.



You should use this validation service to check and correct any web page that you design. The easiest way to use the service is to simply click on the Browse button on the page, and select the HTML file that you wish to check. Once the file has been selected, click on the Check button to validate the code. If the page uses correct HTML code, then you should see something similar to the following:


[Valid] Markup Validation ...
validator.w3.org/check
Google
Congratulations
The uploaded document "example.html" was successfully checked as HTML5. This means that the resource in question identified itself as "HTML5" and that we successfully performed a formal validation of it. The parser implementations we used for this check are based on validator.nu (HTML5).
Validating CSS Style Sheets
If you use CSS in your document, you can check it using the W3C CSS Validation Service.
↑ TOP
Home About... News Docs Help & FAQ Feedback Contribute
This service runs the W3C Markup Validator, v1.3.
COPYRIGHT © 1994-2012 W3C® (MIT, ERCIM, KEIO), ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH THIS SITE ARE IN ACCORDANCE WITH OUR PUBLIC AND MEMBER PRIVACY STATEMENTS.
I ♥ VALIDATOR

5.11 Quick Reference List

The following chart lists the tags we have covered in this document.

Essential tags	Purpose
`<!DOCTYPE>`	The type of the document (DTD)
`<html>`	The entire HTML document
`<head>`	The head of the document (information for the browser)
`<title>`	The title for the window (also used for navigation)
`<body>`	The body of the document (the page content)

Basic formatting tags	Purpose
`<h1>` to `<h6>`	Headings level one to six
`<p>`	A paragraph
` `	A line break
`<hr>`	A horizontal rule
`<pre>`	Preformatted text

List tags	Purpose
`<ul>`	An unordered list
`<ol>`	An ordered list
`<li>`	A list item
`<dl>`	A definition list
`<dt>`	A definition term
`<dd>`	A definition description

Table tags	Purpose
`<table>`	A table
`<tr>`	A row in the table
`<td>`	A cell within the row

Special tags	Purpose
`<!-- ... -->`	A comment

Link tags	Purpose
`<a>`	An anchor (hypertext reference)

Image tags	Purpose
`<figure>`	Group image and caption
`<img>`	An image
`<figcaption>`	A figure caption

HTML5 Non-Semantic	Purpose
`<div>`	Division
`<span>`	Group inline-elements
`<section>`	Defines a section

Video	Purpose
`<video>`	A video

5.12 References

- http://www.w3schools.com/html/
- http://validator.w3.org/
- http://www.w3.org/Style/LieBos2e/enter/

CHAPTER 6

Cascading Style Sheets (CSS)

6.1 Introduction

All the tags we have considered so far have been used to define the structure of the document. To change the visual appearance of any of those structures, we use CSS or Cascading Style Sheets.

The idea underlying CSS is that we use the XHTML tags to define the structure of the document, then we apply a style to define the visual appearance of each structure. This approach keeps the structure of the document separate from the appearance.

6.2 Style definitions

All styles defined using CSS have the same general format:

```
selector {property: value;}
```

selector The selector defines the element that the style will be applied to. It is normally an XHTML tag such as `p` or `h1`.

property The property is used to select the visual aspect of the element you want to change. For example, you might want to change the `color`, the `text-align` or the `font-size`.

value The property is changed to the value specified.

In the example below, the style is used to change the appearance of all first-level headings (`<h1>` tags), since the selector is `h1`. The colour of the first-level headings is changed, since the property to change is the `color`. Since the value we apply to

the `color` property is `green`, the first-level headings in this document will be coloured green.

```
h1 {color: green;}
```

6.2.1 Changing multiple properties for a selector

We can apply more than one change to a given selector. For example, the following style definition will centre all the paragraphs in the document and make them coloured red.

```
p {text-align: center;}
p {color: red;}
```

Although the code shown above is valid, it is usual to include the changes to a given selector within the same curly braces as follows:

```
p {text-align: center; color: red;}
```

It is easier to read the styles if each property is listed on a separate line, so the style shown above would normally be written as follows:

```
p
{
text-align: center;
color: red;
}
```

This is the preferred way to apply both of the changes to the paragraph text.

6.2.2 Defining a style that has multiple selectors

If we want to apply the same style to a number of different elements on the page, then we could define a style for each of the selectors. For example, imagine that we want all of the headings on a page to be blue and aligned to the left. We could use the following styles:

```
h1 {color: blue; text-align: left;}
h2 {color: blue; text-align: left;}
h3 {color: blue; text-align: left;}
h4 {color: blue; text-align: left;}
h5 {color: blue; text-align: left;}
h6 {color: blue; text-align: left;}
```

Alternatively, we could use the following code:

```
h1, h2, h3, h4, h5, h6
{
color: blue;
text-align: left;
}
```

We are allowed to list multiple selectors (each separated with a comma) and apply the same style to all of them at once. This is preferable to the first example because it makes it obvious that all the selectors are supposed to use the same style.

6.2.3 The class selector

We can define a "class" selector that will be used to apply a style to all tags that belong to that class. For example, we might be writing a web page where we want some of the headings to be blue, and some headings to be red. We also want some paragraphs to be blue and some to be red. We can easily achieve this using a class selector.

A class selector always starts with a full-stop. Following the full-stop, we choose a name for our class, then we define the style that will be used by elements that belong to that class. The format is as follows:

```
.className { property: value; }
```

In the XHTML source code, we specify that an element belongs to the class using the syntax:

```
<tag class="classname"> ... </tag>
```

For example, the following styles are used to create two different classes, one for the colour blue and one for the colour red.

```
.hot {color: red;}
.cold {color: blue;}
```

The XHTML source code that uses these classes might look something like the following:

```
<h2 class="hot">The Sahara Desert</h2>
<p class="hot">
Temperatures in the Sahara regularly exceed
50 degrees centigrade
</p>

<h2 class="cold">Antarctica</h2>
<p class="cold">
Temperatures in Antarctica are extremely low
</p>
```

The class selector is used when we want to apply the style to more than one tag, in other words, we want to create a new group and apply the style to all the elements in that group.

6.2.4 The id selector

We can define an "id" selector that will be used to apply a style to a single tag with the specified id. For example, we might have a style change that we want to apply only once. This could be done using an inline style (see 6.3.3), but using an id selector allows us to define all the styles in the same place.

An id selector always starts with a hash sign (#). Following the hash, we choose a name for the id, then we define the style that will be used by the element that has the id in question. The format is as follows:

```
#idName { property: value; }
```

In the XHTML source code, we specify that an element belongs to the class using the syntax:

```
<tag id="idName"> ... </tag>
```

For example, the following style colours the text yellow and aligns it to the right side of the page.

```
#mainHeading {color: yellow; text-align: right;}
```

The XHTML source code that uses this style might look something like the following:

```
<h1 id="mainHeading">Bananas</h1>
<p>A banana is a fruit that has a yellow skin.
Monkeys are often portrayed eating bananas.</p>
```

6.2.5 Other selectors

A number of other selectors are also supported. These are used to apply styles when the user interacts with the element.

`:active`

The `:active` pseudo-class is used to apply a style when an element is active. This occurs when the user clicks the mouse on the element. It is used in the following way:

```
tag:active
{
   ... style is defined here
}
```

For example:

```
h1:active
{
    background-color: blue;
}
```

:hover

The :hover pseudo-class is used to apply a style when the user moves the mouse over the element. This is frequently applied to the a tag to make links that change when the user moves the mouse over them (e.g. in the case of buttons that act as links). It is used in the following way:

```
tag:hover
{
   ... style is defined here
}
```

For example:

```
a { background-color: green; }
a:hover { background-color: lime; }
```

:link

The :link pseudo-class defines the style that is used by a hypertext reference before the link has been followed (i.e. it defines the style of an unvisited link). It is used in the following way:

```
a:link
{
   ... style is defined here
}
```

:visited

The :visited pseudo-class defines the style that is used by a hypertext reference after the link has been followed (i.e. it defines the style of an visited link). It is used in the following way:

```
a:visited
{
   ... style is defined here
}
```

6.3 Location of styles

A style can be defined in three different locations. These are outlined in this section.

6.3.1 An external style sheet

An external style sheet is used when we want to apply the same styles to a number of different web pages. This is useful when we want to maintain a consistent visual theme throughout an entire website.

To use an external style sheet, we simply define all the styles in a separate file. In the source code for our web page, we add a tag that tells the browser that we are using the styles in the separate file.

For example, we could call the file that contains the styles `mystyles.css`. The styles are written in plain text as follows:

```
h3 {color: blue; text-align: left;}
... other styles included
```

The document containing all the content (the XHTML web page) would refer to that style sheet using the `<link>` tag as shown below:

```
<head>
<title>A page that includes a style sheet</title>
<link rel="stylesheet" href="mystyles.css"
 type="text/css"></link>
</head>
```

Note that the `<style>` tag is not required in either file when this method is used.

6.3.2 An internal style sheet

If the styles will only apply to content appearing in a single web page, then an internal style sheet should be used.

This approach uses a `<style>` tag within the `<head>` of an XHTML document. All the styles that apply to that page should be contained within the `style` tag. For example:

```
<head>
<title>A page that includes a style sheet</title>
<style type="text/css">

   ... styles are defined here
```

```
</style>
</head>
```

This is the approach that would normally be taken when a single web page was developed.

6.3.3 An inline style

A style can also be applied to an individual tag. This can be useful when there is a change that will be made to a single tag, although this approach should be used sparingly. It is normally better to have all the style changes listed in an internal style sheet, keeping the visual appearance defined in an area which is distinct from the content.

For example, the following code shows how a style can be applied to an individual paragraph:

```
<p style="color: green;">This paragraph is green</p>
```

6.3.4 Applying styles in order

The system is called Cascading Style Sheets because styles defined in one place can override the styles defined elsewhere according to a hierarchy. The styles "cascade" over each other to form the final style sheet that will be used on a given page.

If no styles are defined, then the Browser default styles will be used. If styles are defined in an external style sheet, then those styles will override the Browser styles. If styles are defined in an internal style sheet, then they will override the Browser defaults and any externally defined styles. If an inline style has been defined for an individual element, then it will override all the other styles. The hierarchy is then listed as follows (with the lowest priority style listed first).

1. Browser default
2. External style sheet
3. Internal style sheet
4. Inline style

6.4 and

There are two additional tags that are used in conjunction with CSS. These are and . They have no inherent display properties. In other words, if you use a or tag to enclose your content and use the browser defaults, then it will not affect the visual appearance at all.

However, these tags can be used to group elements or text in the content of the page. A style can be applied to the tag to cause a desired visual effect. For example, if the

following code was displayed on a web browser, then the and tags would have no effect on the appearance.

```
<div>
<p>This is a very <span>short</span> paragraph.</p>
<p>This is another short paragraph.</p>
</div>
<p>Both paragraphs above are contained within the same
div block, so a style could be applied to them and
would affect both paragraphs.</p>
```

However, if we created a style that applied to those tags, then it would cause the appearance to change. We will use the style sheet defined as follows:

```
.red { color: red; }
.important { font-weight: bold; }
```

We can use these styles to change the appearance of elements contained within the and tags as follows:

```
<div class="red">
<p>
This is a very <span class="important">short</span>
paragraph.
</p>
<p>This is another short paragraph.</p>
</div>
<p>Both paragraphs above are contained within the same
div block, so a style could be applied to them and
would affect both paragraphs.</p>
```

Since the tag contains both paragraphs, all the text in both paragraphs will be coloured red. The text contained within the tag will be coloured red and will also be bold.

The tag is a block-level tag. It may contain other block-level tags, including other tags. The tag is an inline tag. It may contain other inline tags, but may not contain other blocks.

6.5 Properties

The following tables list the different properties that can be used in Cascading Style Sheets. Some properties have been omitted from the tables provided in this section. For a full list, see one of the referenced web sites.

6.5.1 Font

Defines the style used to display the font. The name of the font, weight and style can be set using these properties.

Property	*Description*	*Value*
font-family	Requires a prioritized list of fonts, separated with commas. Font names that include white space must be quoted. E.g. “Zapf Chancery”	*family-name* *generic-family*
font-size	Specifies the size of the font	xx-small x-small small medium large x-large xx-large *length*
font-style	Specifies the style of the font	normal italic
font-variant	Either normal or in a small-caps font	normal small-caps
font-weight	Sets the weight of the font	normal bold

Font family-name

A font family is the name of a font such as “Times”, “Arial” etc. In CSS, these should be enclosed in quote marks. A prioritized list of font family names would be a list from the highest priority to the lowest priority. Each item in the list should be enclosed in quote marks and separated by commas.

Font generic-family

A generic font family is provided as a default option if the preferred fonts cannot be used for some reason (e.g. they are not installed on the system used to read the web page). The generic font families are “serif”, “sans-serif”, “cursive”, “fantasy” and “monospace”. The generic font families are listed below with some of the common font families that belong to each category.

serif

- Times New Roman
- Garamond
- Palatino

sans-serif

- Arial
- Helvetica
- Verdana
- Century Gothic

cursive

- Comic Sans MS
- ITC Zapf Chancery

fantasy

- Cottonwood

monospace

- Courier
- Courier New

Note: If a font style is applied as an inline style, then single quote marks should be used to surround the font name, since double quotes will signify that the end of the `style` attribute has been reached. For example:

```
body { font-family: "New Century Schoolbook", serif }

<body style="font-family: 'My own font', fantasy">
```

Font Example

```
p
{
    font-family: "Verdana", "Helvetica", sans-serif;
    font-size: x-large;
    font-weight: bold;
    font-style: italic;
}
```

6.5.2 Background

Defines the background used for an element. The background can be a simple colour, or could contain an image. The background properties related to images are omitted.

Property	*Description*	*Value*
background-color	Sets the background colour of an element	*color*

6.5.3 Text

The text styles are used to define the way that text is displayed.

Property	*Description*	*Value*
color	Sets the colour of the text	*color*
text-align	Aligns the text within an element	left right center justify
text-indent	Indents the first line of text in an element	*length*
text-transform	Transforms the text in an element	none capitalize uppercase lowercase

6.5.4 Borders

Defines the borders that surround an element. Borders can be set individually for the top, left, bottom and right sides of the element. Properties that apply a border around all sides are also included for convenience.

Property	*Description*	*Value*
border-color	Sets the colours of all four borders	*color*
border-style	Sets the style used for all four borders	none dotted dashed solid double groove ridge inset outset
border-width	Sets the style used for all four borders	thin medium thick *length*

Property	Description	Value
border-bottom-color	Sets the colour of the bottom border	as for border-color
border-bottom-style	Sets the style of the bottom border	as for border-style
border-bottom-width	Sets the style of the bottom border	as for border-width
border-left-color	Sets the colour of the left border	as for border-color
border-left-style	Sets the style of the left border	as for border-style
border-left-width	Sets the style of the left border	as for border-width
border-right-color	Sets the colour of the right border	as for border-color
border-right-style	Sets the style of the right border	as for border-style
border-right-width	Sets the style of the right border	as for border-width
border-top-color	Sets the colour of the top border	as for border-color
border-top-style	Sets the style of the top border	as for border-style
border-top-width	Sets the style of the top border	as for border-width

6.5.5 Table Borders

When using a table with HTML5, to define table borders you need to use CSS. This is done using the **border** property which needs to be specified for the `<table>`, `<tr>` and `<td>` elements. An example specifying a 1 pixel wide, solid black border is given below:

```
table, tr, td
{
border: 1px  solid  black;
}
```

6.5.6 Lengths

A length must always be a numeric value, followed by the unit of measurement. The length can be specified in a number of different units as shown in the following table:

Unit	Description
cm	centimetre
mm	millimetre
pt	1 pt is $\frac{1}{72}$inch. Often used to describe size of fonts.
px	pixels

For example, a length could be `1.5cm` or `32mm` or `100px`.

6.5.7 Colours

Colours may be represented using a colour name, a hex value, or an rgb value. Only sixteen different colours can be used by name. For any other colours, a hex value or rgb value should be used. Each of the colour formats are listed below:

Colour Name

- aqua
- black
- blue
- fuchsia
- gray
- green
- lime
- maroon
- navy
- olive
- purple
- red
- silver
- teal
- white
- yellow

Hex value

A colour may be specified using a hexadecimal form. This form consists of a hash sign (#), followed by a 6 digit hexadecimal number. The first two digits of the number represent the amount of red, the next two digits represent the amount of green, the last two digits represent the amount of blue.

Hexadecimal numbers use the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, a, b, c, d, e, f. The letters a–f represent the decimal values from 10 to 15 (so "ac" in hexadecimal is 10*16 + 12 = 172). Since this form is the common way for colours to be represented in HTML, there are full colour charts that show the colours corresponding to each hexadecimal value. One such colour chart is located at:

http://www.w3schools.com/html/html_colors.asp

A few examples of hex values and the corresponding colours are listed below:

```
#000000 - black
#ffffff - white
#ff0000 - red
#00ff00 - lime
#0000ff - blue
#800080 - purple
```

RGB value

A colour may also be specified using an rgb value in decimal. In this format, the text `rgb( )` are used. Within the parentheses, three decimal values (within the range 0–255) are used to indicate the amount of red, green and blue colour respectively. The previous hexadecimal colours would be represented using an rgb value as follows:

```
rgb(0,0,0)       - black
rgb(255,255,255) - white
rgb(255,0,0)     - red
rgb(0,255,0)     - lime
rgb(0,0,255)     - blue
rgb(128,0,128)   - purple
```

Example

The following example shows how different colour values are used in CSS.

```
body {color: black; background: white; }
h1 { color: #008080; }
h2 { color: rgb(128,128,0); }
```

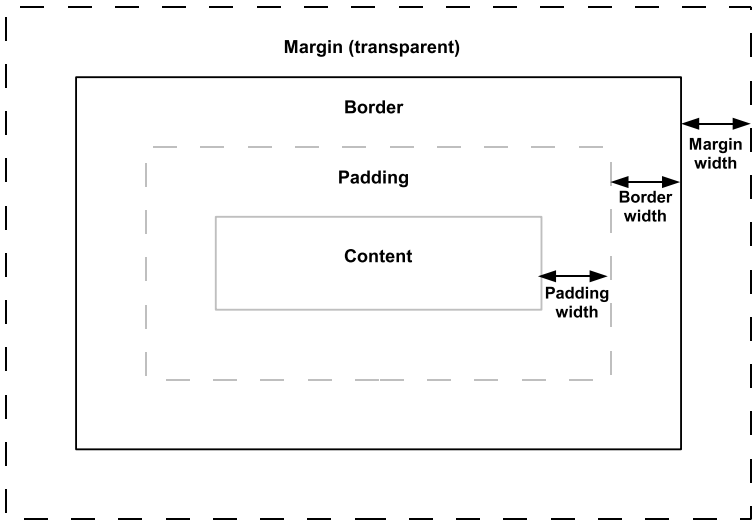
6.6 Advanced CSS (not examinable)

The system used to position elements on the page is the most complicated part of CSS. If you want to use these advanced features then you will have to experiment a lot. The pages you create can look very nice, so the effort pays off in the long run, but it can initially be quite confusing. Using these properties is not recommended unless you are very confident with the properties described previously.

The box model used to position the different elements of the page is described below:

6.6.1 Box model

The box model defines the way that margins, padding and content are used for layout. The following diagram explains how these are interpreted in this model.



The content area contains the text or other content of the element (including any further elements contained by this one). The padding area is coloured the same as the background of the content. The border colour is set using the border properties. The margin area is transparent, and is used to separate this element from other elements.

It is important to note that vertical margins will be collapsed where they overlap. In other words, if the margins for two elements overlap vertically, only the larger margin will be used.

6.6.2 Padding

Padding is inserted around the content to provide some additional space between the content and the surrounding. It is coloured the same as the background of the content. See section 6.6.1 for more information.

Property	*Description*	*Value*
padding	Sets the top,left, bottom and right padding	*length*
padding-bottom	Sets the bottom padding of an element	*length*
padding-left	Sets the left padding of an element	*length*
padding-right	Sets the right padding of an element	*length*
padding-top	Sets the top padding of an element	*length*

6.6.3 Margins

Margins are transparent. If a margin is set for an element, then the colour of those margins is determined by the enclosing element (i.e. since the margins are transparent, you can see through them to the element behind). Margins can be set to negative values. See section 6.6.1 for more information.

Property	*Description*	*Value*
margin	Sets the top, left, bottom and right margin	auto *length*
margin-bottom	Sets the bottom margin of an element	auto *length*
margin-left	Sets the left margin of an element	auto *length*
margin-right	Sets the right margin of an element	auto *length*
margin-top	Sets the top margin of an element	auto *length*

6.6.4 Positioning

These define how the entire element is treated on the page, in terms of visibility and positioning.

Property	*Description*	*Value*
float	Specifies where the element will appear with respect to the enclosing element	left right none
position	Defines how the element is positioned	static relative absolute fixed
display	Sets how the element is displayed	block inline none
top	Sets the distance to the top of the element. Not used if the position is static	*length*
left	Sets the distance to the left side of the element. Not used if the position is static	*length*
bottom	Sets the distance to the bottom of the element. Not used if the position is static	*length*
right	Sets the distance to the right side of the element. Not used if the position is static	*length*

6.6.5 Dimension

These define the height and width of elements on the page.

Property	*Description*	*Value*
height	Sets the height of an element	auto *length*
max-height	Sets the maximum height of an element	none *length*
min-height	Sets the minimum height of an element	*length*
width	Sets the width of an element	auto *length*
max-width	Sets the maximum width of an element	none *length*
min-width	Sets the minimum width of an element	*length*

6.7 References

- http://www.w3schools.com/css/default.asp
- http://www.htmlhelp.com/reference/css/all-properties.html
- http://csszengarden.com/

- http://www.brainjar.com/

CHAPTER 7

PowerPoint

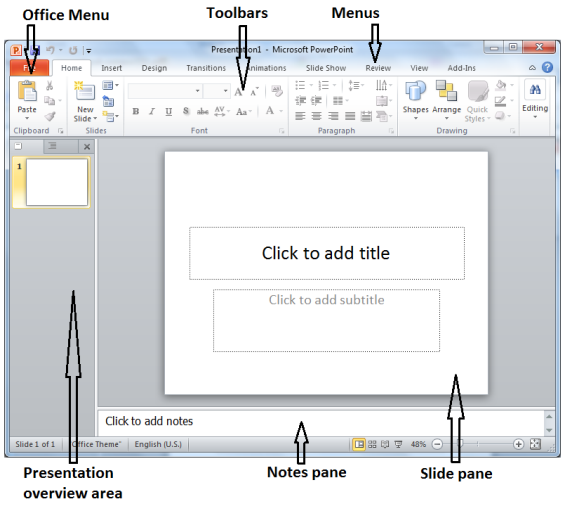
7.1 Overview

PowerPoint is an application used to create presentations. These presentations are arranged as a series of slides. Unlike a word processor, information added to a slide does not “wrap” from one page to another, rather in PowerPoint, each slide is created independently from the other slides.

Although PowerPoint is most often used to support a spoken presentation, it can equally be used to create a presentation that runs independently at a kiosk or information centre.

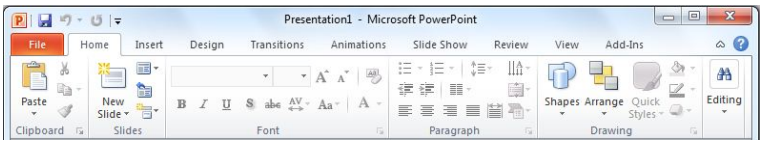
7.2 Getting started

When you start PowerPoint, you will see a window similar to the one shown in the following diagram. The main elements of the PowerPoint interface will be discussed in this section.

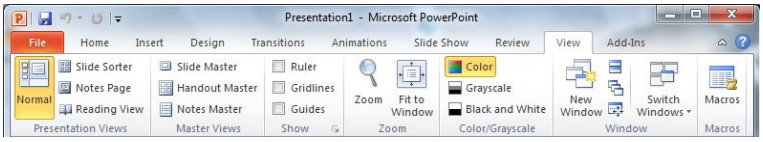


Menus and Toolbars

Office 2007 and 2010 do not use menus in the same way as previous versions. The main menus are now presented as a tabbed Toolbar. In other words, the items are no longer chosen by picking from a list, but rather they are presented as a Toolbar (this new toolbar is called a "ribbon" by Microsoft). For example, the Home menu results in the Home tab being displayed:



while the View menu results in the View tab being displayed.



TIP: If you hover the mouse over any of the Toolbar buttons then a pop-up box will appear explaining what the button does.

Presentation overview

On the left side of the window is an area that gives you an overview of your presentation. It contains two different tabs, the Outline tab and the Slides tab. The Slides tab contains thumbnails (small pictures) of all the slides that you have created for the current presentation. You can use these thumbnails to navigate through your presentation (i.e. to move to a particular slide).

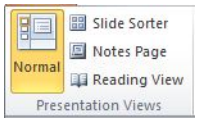
Notes Pane

Each slide in a PowerPoint presentation has an associated Notes area. These notes are not displayed as part of the presentation, but you can print them out if you wish. This is the ideal place to make your own personal notes about a presentation (perhaps to remind yourself of what you wanted to say about the slide).

7.2.1 Views

Slides that form part of a PowerPoint presentation can be viewed in many different ways. The normal view is used to develop the slides, the slide sorter view allows you to rearrange slides easily, the notes view allows you to create notes for the slide, and the slide show view is used to display the presentation in its final form.

The different views can be accessed using the View tab. The views are grouped together in the area labelled Presentation Views.



The Normal view is used to develop the slides. It is the default view that you see when you first start the PowerPoint application.

Slide Sorter view

Selecting View → Slide Sorter will show all the slides in the presentation. This allows you the opportunity to change the order of the slides (by clicking and dragging to a new position). It is worth noting that you can perform the same task in normal view by clicking and dragging the slide in the Slides tab.

Use this view to determine whether your slides flow naturally and logically from one to the next. You may notice that a slide needs to be broken up into a larger number of slides (perhaps 2 or 3 slides) to improve the structure and clarity of the presentation.

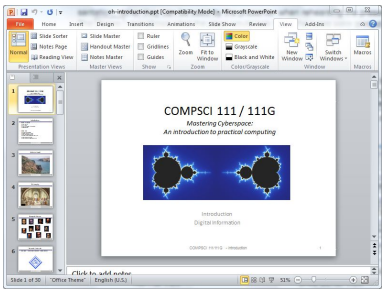
Notes view

Choosing View → Notes Page will display the slide and any notes that are attached to that slide. As mentioned earlier, the notes themselves do not appear as part of the presentation, rather they can be used by the presenter when rehearsing the presentation. The Notes Page view shows what the page would look like if the notes were printed out.

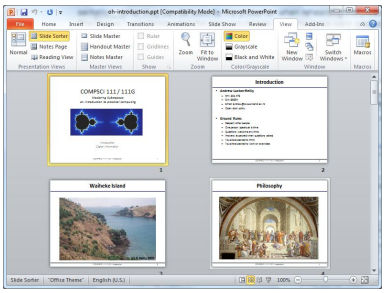
Slide Show view

Choosing Slide Show → From Current Slide will start the presentation from the current slide. If your presentation includes any advanced features such as slide transitions,

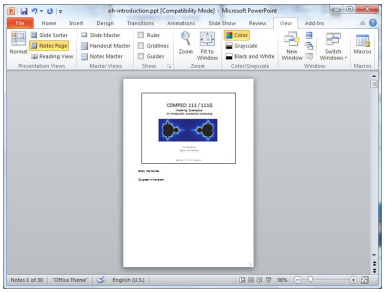
sound or animation, then you can verify that those features work correctly by using the Slide Show view. To return to the normal view from From Current Slide press the Esc key.



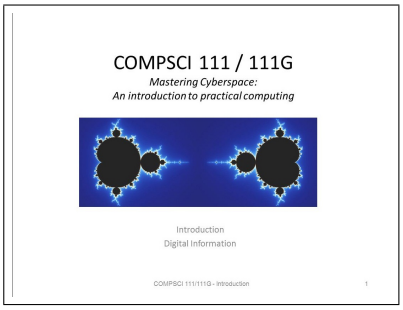
Normal view



Slide Sorter view



Notes view



Slide Show view

7.2.2 Options

PowerPoint has a number of options that allow you to tailor the way various automatic features operate. Don't be afraid to experiment with these options (although it might be worth writing down what you have changed in case you want to change it back). PowerPoint (as with most Microsoft products) is designed to "help" you as much as possible. As you type, spelling mistakes and formatting errors will be automatically corrected. PowerPoint guesses what you are trying to do and changes the formatting accordingly. For example, if you start a line with an asterisk like so:

```
* this is a sentence
```

then PowerPoint will automatically change the line into a bullet pointed list and the first word of the sentence will be capitalized.

- This is a sentence

Although this is sometimes helpful, there are times that you want to display your information in a specific way and you do not want any interference.

The File botton will bring up a window that contains a button called Options. Clicking this button allows you to configure the various options that control the level of automation used by PowerPoint.

7.3 Adding content

PowerPoint presentations consist of a title slide and one or more slides containing the content.

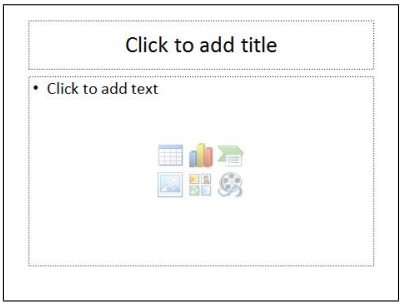
7.3.1 Title slide

The title slide is normally the first slide of the presentation. The default layout for the title slide is different than the other slides. When you start a new blank presentation, you should see something similar to the following:

The title slide contains placeholders for the main title and a subtitle. To add titles, click in the appropriate boxes and enter your text.

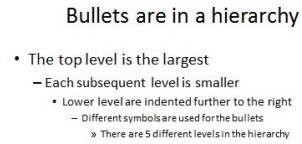
7.3.2 Adding a new slide

When you add a new slide to the presentation, it is inserted *after* the slide that is currently selected. To insert a new slide to your presentation, you can click on a slide (or between two slides) in the Slides tab in the presentation overview area and hit the Enter key. In either case, a new slide will be inserted into the presentation and you can start to add content to that slide. The new slide will contain placeholders for the slide title and the slide content area. Simply click in the appropriate boxes and enter your text.



7.3.3 Bullet Points

PowerPoint is designed to present text as bullet points. These bullet points are organised in a hierarchy. The top-level points are large, bold, and appear on the left side. As the points move lower in the hierarchy, the text appears smaller and is indented further to the right.



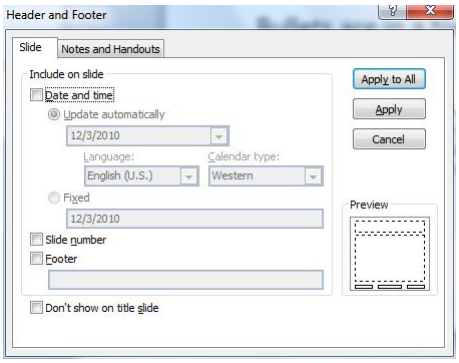
Click on the button  to increase the level of indentation (lower in the hierarchy) or the button  to decrease the level of indentation (higher in the hierarchy).

Both buttons are located on the Paragraph section of the Home Toolbar.

7.3.4 Headers and Footers

A header or a footer can be added to the printed pages when you print the slides as handouts (or as notes). However, the slides themselves may only have a footer when they are displayed on the screen.

The header/footer may contain the date (either updated automatically to the date you last modified the slide, or a fixed date of your choice), the slide number, or any fixed text that you specify (such as the name of your department/company).



A different footer can be applied to different slides, although it is more common to have a consistent footer on all the slides. Note that the title page can have a different footer.

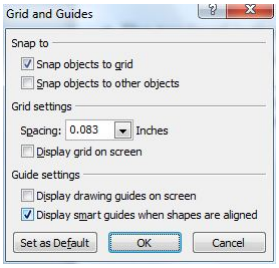
You can add information to the header or footer of a slide by choosing the Insert tab → Header & Footer.

7.3.5 Drawing tools

The drawing tools provided with PowerPoint are very powerful and easy to use. If you intend to use diagrams, flow charts or pictures on your slides, then you need to learn how to use the drawing tools.

Grids

PowerPoint uses the idea of a drawing grid that helps you to align drawing objects more easily. Imagine that the screen is covered by an invisible grid, and when you create or move a drawing object, the edges of the object will always be aligned to one of the grid lines (i.e. the object will "jump" in size or location from one grid line to the next). You are able to set the size of the grid to a value that is suitable for the drawing that you plan to create (a value between 0.05cm and 1.0 cm is fairly typical).



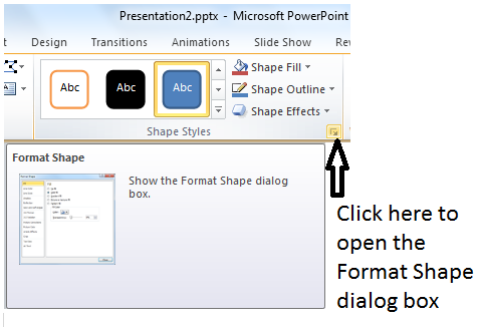
From the Home tab, click on the Arrange icon and choose Align → Grid Settings

Working with shapes

PowerPoint has many pre-defined shapes that you can choose from. Once you select the shape you want to draw, then clicking and dragging the mouse will draw the chosen shape on the slide. After the shape is drawn, you can alter various properties such as the outline and fill.

Shapes can be chosen from the Insert tab, by clicking on the Shapes button. To change the properties of the object, select the object (by clicking on it) and choose the Drawing Tools – Format tab. Alternatively, double-clicking the shape will automatically select the Drawing Tools – Format tab.

The "Shape Style" section of the tab allows you precise control over the appearance of the selected object.



TIP Using the Connector shape creates a "sticky" connection between two objects. If one of the objects is moved to a new position then the end of the connector will remain attached. This is a very useful feature when creating flow charts and diagrams that use lines to connect various parts.

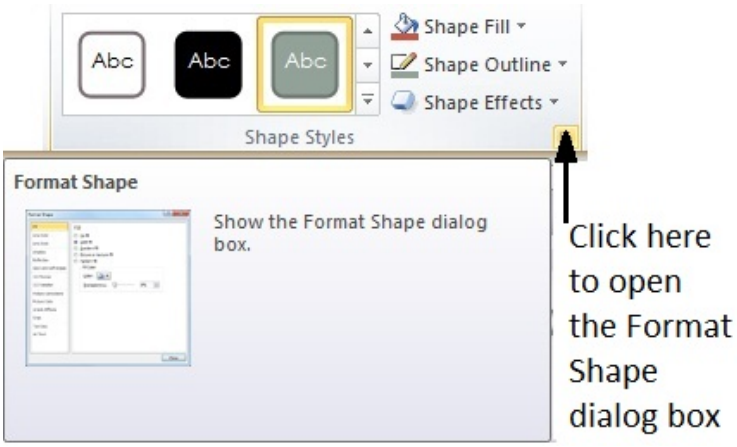
Each object (including any text boxes on the slide) floats in a different layer (imagine that each object is drawn onto a separate pane of glass and all of the panes are stacked up into a big pile). We can alter the order of these layers which in turn defines which objects appear in front of other objects.

To change the ordering of objects, choose Drawing Tools – Format then select the appropriate button from the "Arrange" section of the tab.

Objects can also be rotated or flipped (like a mirror image) using options from the menus described above. If you select a number of shapes, you have the option to Group the objects together (so that they are treated as a single object thereafter), or to Align the objects.

If you want to add text to a shape, simply start typing when that shape is selected. You can change the properties of the text by selecting it and using the normal menus to alter the appearance of the font.

If you want to alter the way the text is anchored to the shape, then use the Format Shape dialog box and select Text Box from the list of options.



7.3.6 Pictures

Pictures can easily be added to PowerPoint presentations using either the clipboard, or by accessing the picture directly from disk.

Adding pictures using the clipboard

The most common way to add pictures to your PowerPoint presentations is using the clipboard. If you are viewing the image using your web browser, you can right-click the image and choose Copy Image or Copy (depending on your browser). If you are using another application to view the image, then you can probably choose Edit → Copy. Once the image is copied to the clipboard, select the PowerPoint slide you want and paste the image from the clipboard.

To paste a picture from the clipboard, choose the Home tab, then Paste from the "Clipboard" section of the tab.

Adding pictures that are stored on disk

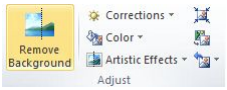
Alternatively, you can insert a picture directly from your disk (useful for adding photos) onto a slide.

Click on the Insert tab, then choose Picture from the Images section of the tab.

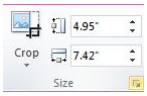
Simple editing of pictures

You can make simple changes to pictures from within PowerPoint. For more complex changes, you would need to use a specialist photo editing product. However, PowerPoint is capable of adjusting the brightness and contrast, and can be used to crop the image.

Selecting a picture will reveal the Picture Tools–Format tab, which contains all the tools used to alter the picture. The "adjust" section allows you to alter the brightness, contrast or colour of the image.



The "size" section allows you to alter the size, and to crop the image.



Compressing images

Images take up a lot of memory on your disk. A PowerPoint presentation without any images would typically take a small amount of memory, perhaps 100KB. This is certainly small enough to be stored or emailed easily.

A presentation that contained a number of pictures could easily take more than 10MB of space (more than 100 times as much as the presentation without images). Such a large presentation is difficult to transfer via email and may be difficult to maintain on portable media if many such presentations are created.

PowerPoint includes a feature that allows you to compress the images belonging to your presentation in order to reduce the overall size of your file. Compressing all the images in a PowerPoint presentation can reduce the size to a quarter of the original size. This is only really necessary for presentations that use a large number of images (or that use very high quality images).

Use the Compress Pictures button in the "Adjust" section of the Picture Tools–Format tab.

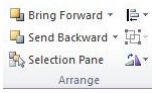
Ordering the layers

Since a picture is treated as a special kind of object, many of the standard tools used for drawing can also be applied to pictures. The layer of a picture can be shifted towards the front or back, the picture can be rotated, aligned and distributed, just as we can do with drawing objects.

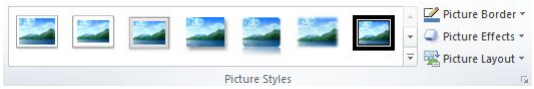
For example, you can create a black outline around the picture by choosing to alter the outline property.

You can also make a picture into a background image by making it the back layer. This is an easy way to introduce background images into selected slides (although if you want the same background on all your slides then you should define a new slide design using a Slide Master as discussed in section 7.4.6).

The "Arrange" section of the Picture Tools–Format tab allows you to change the order, alignment, distribution and rotation of your pictures.



The "Picture Styles" section allows you to apply a number of effects to the image, including changing the border and shape of the image.



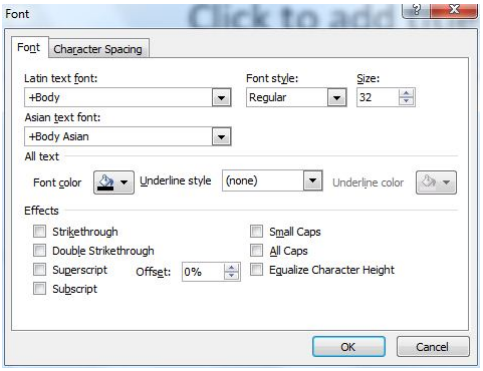
7.4 Making beautiful slides

In the previous section we looked at how to create and view simple presentations. In this section, we will look at ways to make the presentation more attractive.

7.4.1 Formatting text

The tools for formatting text in PowerPoint are similar to those used in word processing applications. You can apply multiple changes to the text one attribute at a time by clicking on the appropriate formatting icon in the toolbar, or you can change many attributes at the same time using the font formatting dialog box (e.g. you could make the text appear in Arial font typeface with 24 point size, bold, italic and underlined).

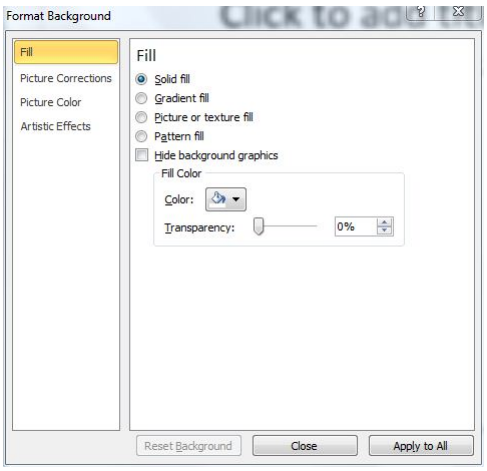
Choose the Home tab and click on the icon that displays the font dialog box in the "Font" section of the tab.



7.4.2 Background and Font colour

It is easy to add a background colour to your slides.

Choose the Design tab, and click on the icon that displays the "Format Background" dialog box. Choose the colour for a simple fill.



Rather than choosing a plain colour for the background, a subtle gradient of colour is often more appealing.

From the “Format Background” dialog box, choose the “Gradient fill” or “Picture or texture fill” option to define the appearance of more complex backgrounds.

If you change the background to a dark colour, then you will have to change the font from the default black colour to something lighter. Use the normal font formatting tools to change the font colour to something that is clearly visible on the background.

7.4.3 Design Theme

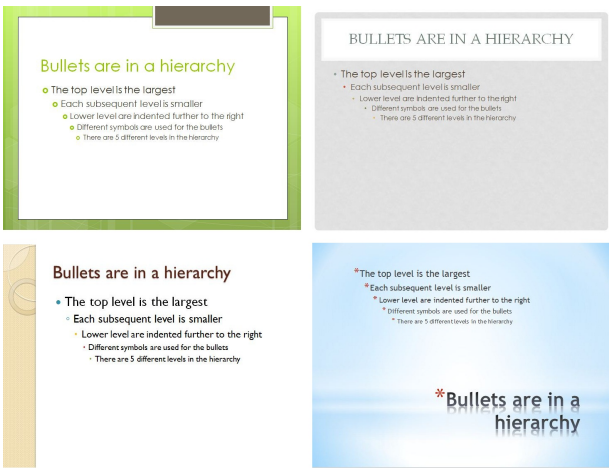
A design theme affects the appearance of your entire presentation. It normally specifies a particular background, the position and appearance of the titles, and the position and appearance of the main content text. PowerPoint comes with a range of built-in templates that you can use. These are called “Themes”.

TIP To create a professional looking presentation, you should try to develop your own theme (as discussed in section 7.4.6). The themes that are built-in to PowerPoint are frequently used by presenters, so audiences are sometimes bored by a style that they have seen so often.

To apply a theme to your presentation, locate the thumbnails that give you a preview of what the theme will look like when it is applied to your presentation.

Choose the Design tab and use the “Themes” section to pick a theme.

You can apply the design by clicking on one of the thumbnails. If you don’t like the design, simply choose another. Note that the design will affect the font size, so sentences that look good on one design might not look good on another.



Examples of Design Templates

7.4.4 Colour scheme

The colour scheme defines the default colours that are used for the slides in your presentation. If you select a design template, then the colour scheme will be chosen as part of that design template, but you are free to choose a different set of colours if you wish. The colour scheme will affect the following elements of each slide:

- Background
- Text and lines
- Shadows
- Title text
- Fill
- Accent
- Accent and hyperlink
- Accent and followed hyperlink

Choose the Design tab and use the Colors button in the “Themes” section to pick a colour scheme to be used for your theme.

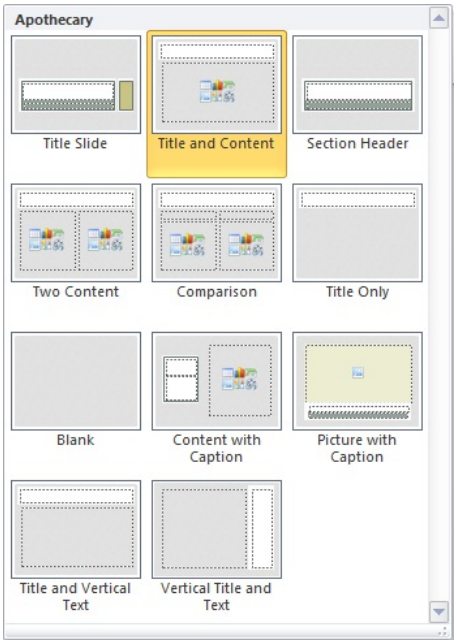
Note that you can specify a different colour scheme for the notes page, the handout page and the slides page within the same presentation. See “Modifying the color scheme” in the *Microsoft Office PowerPoint Help* for more information.

7.4.5 Design layout

A design layout specifies the position of different elements of content within a single slide. You can use a design layout to create slides that have multiple different forms of content (such as charts, pictures, text and diagrams) arranged according to the specified layout. For example, you can use a design layout to have two columns of text, or to have text on the left side and pictures on the right side. This is a quick way to arrange the content on a slide.

Choose the Home tab and select the Layout button from the “Slides” section.

A number of different layouts will be available for you to select from. The thumbnails displayed will give an indication of what different layouts will look like. You can apply the layout by clicking on one of the thumbnails.



Examples of Design Layouts

Different design templates have different amounts of space to use for content (for example, a design template with a very large heading might have less vertical space, whereas a design template with a very large graphic on the left side might have less horizontal space). If you apply a different design template to a slide that has a specified layout, you may have to return to that slide and reset the layout to ensure that the content is displayed correctly.

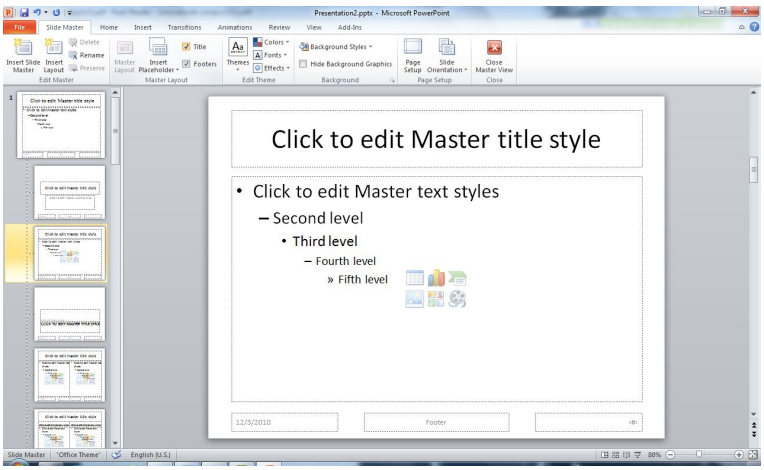
To reset the layout, right-click on the slide and choose Reset Slide.

7.4.6 Using Masters

Masters are used to create a new slide design that can be used as a template for all the slides in a given presentation. In order to create a consistent, professional presentation that has a unique, personalized appearance, you should define your own Slide Master.

To edit the Slide Master, choose the View tab, then choose the Slide Master button from the “Presentation Views” section.

The slide master will be displayed in the main workspace area.



A Slide Master

You can alter the appearance of this Slide Master using any of the standard formatting tools. The size and location of the text boxes, the font style, and the colour can all be altered. The changes you make to this slide master will define the appearance of any new slide that is added to this presentation.

You may also want to define a different template for the title slide, especially if you are giving a series of presentations and you want the appearance of the titles to be consistent in all of them.

When you are finished editing the Masters, you should return to the default view.

Click the Close Master View button in the Slide Master tab, or choose the View tab and click the Normal button.

7.5 Interactivity — animation and multimedia

A number of more advanced features are provided with PowerPoint and it is these tools that are used to make the most compelling presentations.

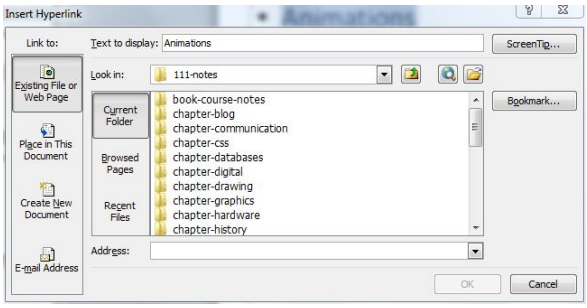
7.5.1 Slide transitions

A slide transition is an animation that occurs when you move from one slide to the next in a presentation. You can apply different slide transitions to different slides if you wish, or use the same transition for all the slides in the presentation. You can change the speed of the transition and you can add a sound that will be played at the same time as the transition. Finally, you can specify whether the slide will automatically advance to the next one after a certain number of seconds has passed, or whether the slide must be manually advanced.

Choose the Transition tab and select the type of transition from the “Transition to this slide” section.

7.5.2 Linking to external resources

You can create a link from a PowerPoint slide to any document that you have on your disk, or to any web page. To add a link, click in a text box on your slide and choose Insert → Hyperlink (or simply right-click in a text box and choose Hyperlink). A dialog box will appear allowing you to select the destination of the link and enter a name for the link.



The Hyperlink dialog box

The name that will appear on the page should be entered in the field entitled “Text to display:” at the top of the dialog box. If you are linking to a document stored on your computer, then use the main area of the dialog box to locate that document. If you are linking to a web site, then copy the address of the web site and paste it into the field entitled “Address” at the bottom of the dialog box. When you are satisfied that the correct information has been entered, click OK to create the hyperlink on the page. You can test the hyperlink by selecting Slide Show → From Current Slide and clicking on the link.

Sound

Sound can be added to your presentation using Insert → Video or Audio from the Media section. If you choose to add sound from a CD, then you must have the CD inserted into the drive each time you deliver the presentation.

TIP If you want to use sounds from your CD, then you can use software to extract the music file to your local hard drive and use that file when you insert the sound into your presentation. Inserting from the file in this way means that the presentation will still work without your CD.

When you insert a sound into your presentation, you will be asked if you want to start the sound automatically (i.e. the sound starts when the slide first appears), or if you want to click to start the sound.

Either way, you will see a small icon of a speaker (or a CD if you have inserted sound from a CD) appear on your presentation when the sound is inserted. This icon repre-

sents the sound that has been added. Don't worry about the icon looking untidy, you can hide the actual icon when you deliver the presentation.

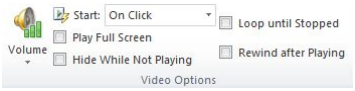
The speaker icon used to represent a sound file

Movies

Movies are complicated because there are so many different kinds of format (the most common are .mpeg, .mov, .avi and .wmv). In addition to the file format, there are a large number of different codecs used to encode the information so that it takes less disk space. A movie that plays on one computer may not play on another if the same codecs are not installed. If you intend to use movies in your PowerPoint presentations then it is essential that you try them out on the machine you will use to deliver the presentation.

Movies are inserted in a similar way to sounds. Select Insert → Video → Video from File, and locate the movie file on your hard disk. As with sound, you will be asked if you want to start the movie automatically (i.e. the movie starts when the slide first appears), or if you want to click to start the movie.

If you select a movie file that has been added to a slide, then two additional tabs called "Format" and "Playback" will appear. The Playback tab contains the options to play the movie full screen, loop until the user stops the movie and to rewind the movie after it has finished playing.

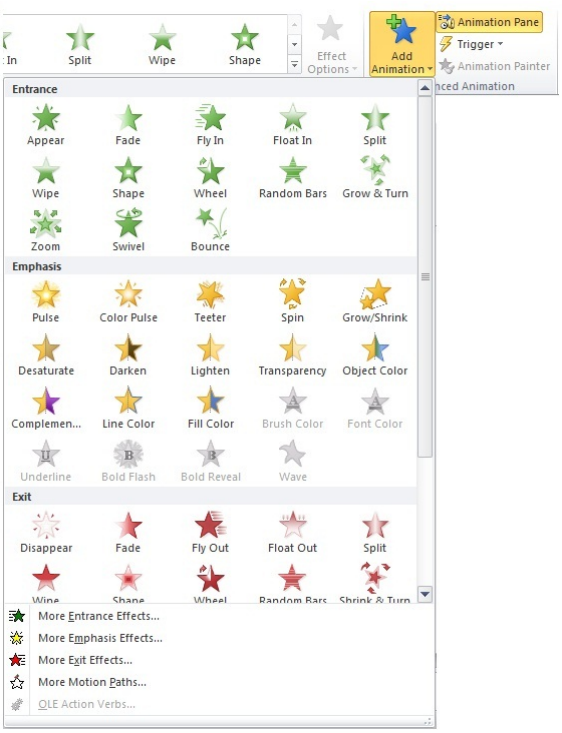


7.5.3 Custom animation

Custom animation allows you to control the animation for each element of your presentation. You can apply custom animation to an individual object or a group of objects. You can animate a single letter, a word, a sentence or an entire paragraph of text.

You can set the triggering conditions for the animation (i.e. when the animation will start). The speed and timing of the animation can be altered, as well as the number of times the animation sequence is repeated. Each animation type has a number of different options that adjusts the way that animation will be displayed.

Custom Animation pane

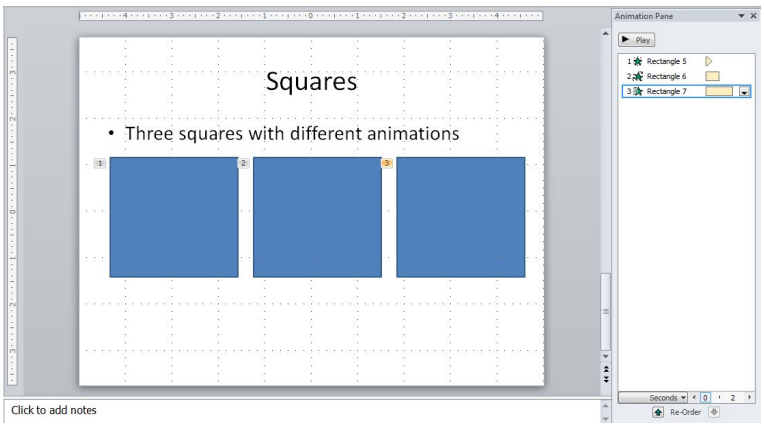


Entrance Animation options

To add a custom animation for an object, first select the object, then choose Add Effect from the Animation tab. You will need to select the type of animation from the following list:

Entrance The animation occurs as the object appears for the first time
Exit The animation occurs when the object is removed from the slide
Emphasis The animation is used to emphasise the object while it is visible
Motion Path The animation is used to move the object from one location to another

Once you have selected the animation you want, you can modify it further in the Animation pane. To display the Animation pane at the right side of the page, click on Animation Pane in the Advanced Animation section on the Animation tab. You will see a small number that appears next to the object in normal view (this will not be displayed during the slide show) which is linked to the item in the Animation pane with the same number.



A slide containing three different animations

Timing

Timing is one of the most important parts of the animation process. You can create complex animations by stringing together a number of simple parts. The timing options allow you to connect animations together with ease. The starting condition for the animation should be one of the following:

On Click The animation starts when the user clicks a button
With Previous The animation starts at the same time as the previous animation (or when the slide first appears if the animation is the first one on the list)
After Previous The animation starts when the previous animation has finished

You can also insert a delay into the timing of the animation by right-clicking the animation in the Animation pane and choosing Timing. Creating effective animation sequences can be time consuming, but ultimately rewarding if you have the time to spend.

Options

Each animation has a number of options that affects the way the animation looks. For example, the spin animation can be set to rotate any number of degrees, either clockwise or anti-clockwise and can be set to slowly speed up and/or slow down at the start and/or end of the spin. These animation options are accessed by right-clicking on the animation in the Animation pane and choosing Effect Options

7.5.4 Package for CD

If you have any linked files (such as movies or sounds) in your presentation then it is critical that you package the presentation for transportation.

Click on the File tab → Save & Send → Package Presentation for CD.

Linked files are *not* embedded in your presentation, instead PowerPoint stores a link to the location of the files on your hard drive. When you copy the presentation to a different

machine (or onto a flash drive), PowerPoint will not automatically update the links, so it will still try to refer to the file on the computer you used to develop the presentation. This means that linked files will not be available during the actual presentation.

To avoid this problem, you can use the option to package the presentation for CD. Although this sounds like you have to store the presentation on a CD, you can actually store the packaged presentation in a folder on your hard drive. The folder will contain the PowerPoint presentation, software that allows you to view the presentation on a computer that doesn't have PowerPoint installed (just in case), and all the files that you linked from the presentation. It is safe to copy this folder to a new location and all the linked files will work correctly.

TIP Always use Package for CD before you move a presentation to a different computer. Always try out your presentation on the destination computer to make sure the fonts, sounds, movies and animation all work as expected before you start to deliver the presentation to an audience.

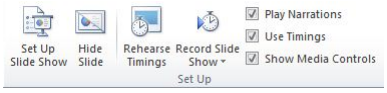
7.6 Presentation

Once you have completed building the presentation, you need to deliver it. This section discusses some of the ways that PowerPoint can be used during the actually delivery.

7.6.1 Rehearsing a presentation

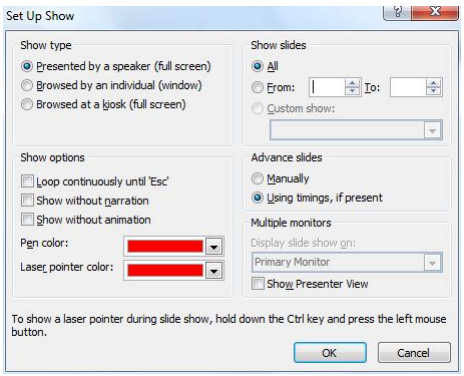
When you have completed the design of your presentation, you may want to rehearse the delivery.

Choose the Slide Show tab, then Rehearse Timings from the "set Up" section.



Your PowerPoint slide show will begin, and you will see a small timer in the top left corner of the screen. As you rehearse your presentation, the timer will record the time when you advance to the next slide. You can pause this timer if you need to take a break. When you have finished the presentation, you can use the Slide Sorter view to see how long you spend on each slide.

You can use the timings generated at this time to automate the playback speed of the slides. If you choose Slide Show → Set Up Show, then you have the option to "Advance Slides" based on the times recorded during rehearsal. If you choose this option then the slides will advance automatically without manual intervention (this is particularly useful if you are setting up a presentation to run at an Information Kiosk).



Setting up the final presentation for delivery

7.6.2 Navigating during a presentation

Once the slide show has started, the slides will either automatically advance or they will have to be manually advanced, depending on your settings (transitions, animations and timings).

You can move through the slides manually in one of the following ways:

- click the left mouse button to advance to the next slide
- press the Space key to advance to the next slide
- press the left or right arrow keys to move to the previous or next slide respectively
- press the number of the slide you want to move to, then press the Enter key (for example, 1-Enter will move directly to the first slide in the presentation).

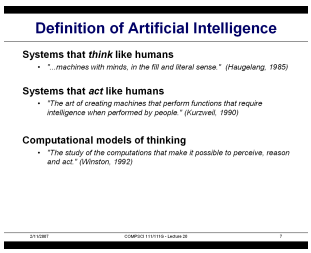
When the presentation begins, the pointer will be invisible. If you move the mouse, then the pointer will appear and you will see some faint buttons appear in the bottom-left corner of the slide. You can click on the left and right icons to navigate to the previous and next slide respectively.

Navigation icons used during a presentation

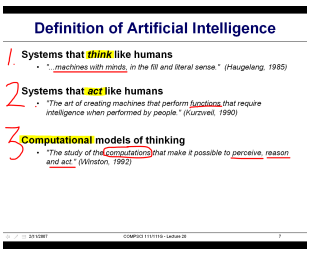
7.6.3 Annotating a presentation

When you are teaching, there are times when you want to highlight elements of a slide, or write on the slide in some fashion. To do this, right click the mouse button and a menu will appear. You can use this menu to choose Pointer Options → either Arrow or Pen or Highlighter. You can also select a colour by right-clicking again and choose Pointer Options → Ink Color and select the colour you want.

Once you have selected a pointer option and a colour, you can scribble on the page using the mouse. When you finish the presentation, you can choose to keep the annotations if you wish.



Before annotations



After annotations

7.7 Design and presentation advice

You can use PowerPoint for different purposes, and the recommendations that apply in one context may not necessarily apply in another. The most important thing is to find a style that works for you (given the time that you have available to spend on the preparation).

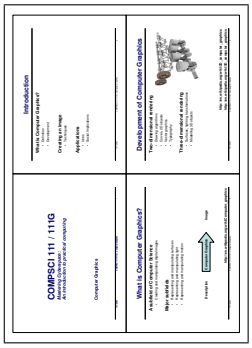
7.7.1 Printing the presentation

There are three main ways that presentations are printed. You can choose to print the slides, the notes page, or print the slides as a handout. The option to select which way to print the presentation is chosen from the Print Dialog box (i.e. choose File → Print as normal and select Handout, Notes or Slides from the dropdown box under the heading "Print what").

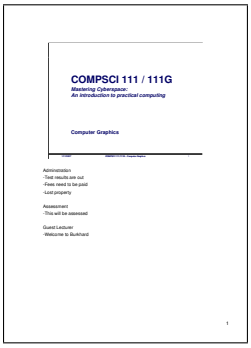
Normally you would choose to print multiple slides on a single A4 sheet if you were using the slides as a handout (perhaps 4 slides per A4 page). The paper should be oriented in Landscape mode for this purpose.

If you choose to print out the Notes page, then it will appear with a single slide per page with any notes located in the space beneath the slide.

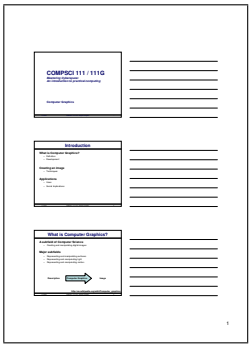
If you choose to print out the presentation as a Handout, then you can select the number of slides that appear on a single page. In the example below, we have chosen 3 slides per page.



Slides (4 per page)



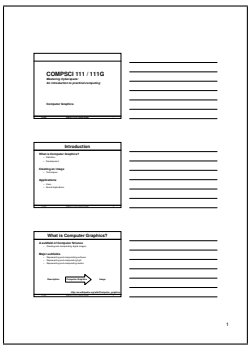
Notes (1 per page)



Handouts (3 per page)

If your slides contain colourful backgrounds then they might be difficult to read when printed. There is an option in the print dialog box to print slides as "'Pure Black and White"'. If you choose this option then the coloured background will be omitted and the text will be printed in black ink to make it easier to read.

Handouts in "Colour" (default)



Handouts in "Pure Black and White"

7.8 Advice on slide design

7.8.1 Your slides support you, not replace you!

The most important thing to remember is that the PowerPoint presentation should support what you say, not replace you entirely. If all the information is on the slides, then you are redundant. It is difficult to read and listen at the same time, so keep the slides simple. The major advantage that PowerPoint brings to a presentation is visual impact, so use it to connect with the audience.

7.8.2 Aim for consistency

The background, colours and fonts should all be consistent throughout your presentation. Style differences between slides are distracting and unprofessional.

7.8.3 Keep it simple

Your slide should contain no superfluous material. Keep only what is essential. The less clutter, the more powerful your message will be.

7.8.4 Limit bullet points and text

The slides should support and enhance your message, not replace you. Prepare a written document that contains your message as a handout and don't use PowerPoint slides as a handout replacement.

7.8.5 Limit animation

Some animation can be good, but keep it simple and quick. Use sparingly. Never use slide transitions! Never!

7.8.6 Limit sound and keep it professional

Sound can be a very powerful medium to carry a message, but use it sparingly. Never use the stock sounds that come with PowerPoint. If you are using music or sound then obtain high-quality sound from a CD source.

7.8.7 Use high-quality visuals

Use stock photography or other high quality images. Avoid clip-art. It cheapens the presentation and makes it look amateurish.

7.8.8 Design your own templates

Don't use the standard Microsoft Templates. They get used too often and have little impact today. Develop your own personalised template.

7.8.9 Make good use of colour

Colour evokes emotion, so choose colours carefully for the feel of the presentation. Never use coloured text for decoration. Use coloured text sparingly and only to emphasise a single word or phrase.

Use warm colours for the foreground (since warm colours tend to stand out) and cool colours for the background (since cool colours tend to recede). Try to maximise the contrast between the foreground and background colours. Limit the number of colours used in a presentation, and use them consistently.

Dark backgrounds with light or white text work best in a dark room. In rooms that are well lit, a white background with dark text is easier to read.

7.8.10 Don't do too much in a single slide

Make sure you have only one main idea per slide. If there is too much for one slide, then split the slide up.

7.8.11 Choose fonts well

Limit the number of fonts that you use in a presentation. No more than two fonts per presentation (e.g. Arial and Arial Bold). Choose a font that is simple and easy to read.

Avoid complicated fonts. Sans-serif fonts are easier to read than serif fonts on screens, so use sans-serif fonts by preference.

7.8.12 Tell a story

A good presentation should engage the audience in a story, not just list a set of facts. The PowerPoint presentation should support and enhance the story that you are telling, not reduce the presentation to a boring list of bullet points. Think of each slide as a short newspaper story. Instead of a heading containing nouns at the top, write a headline that catches your attention. For example, instead of "Problems", write "Poor use of PowerPoint confuses students". Instead of "Advice" write "5 mistakes to avoid".

7.9 References and further reading

- PowerPoint overview
  - http://en.wikipedia.org/wiki/Powerpoint
- Online Training
  - http://office.microsoft.com/en-us/training/default.aspx
  - Microsoft Office Help
- Presentation Zen
  - http://www.presentationzen.com/
  - http://www.garrreynolds.com/Presentation/pdf/presentation_tips.pdf
- Rethinking the Design of Presentation Slides
  - http://www.writing.eng.vt.edu/slides.html
- Stop your presentation before it kills again!
  - http://headrush.typepad.com/creating_passionate_users/2005/06/kill_your_prese.html
- "PowerPoint is Evil" — Edward Tufte
  - http://www.wired.com/wired/archive/11.09/ppt2.html

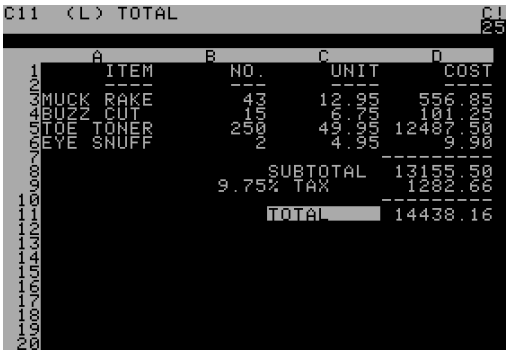
CHAPTER 8

Spreadsheets

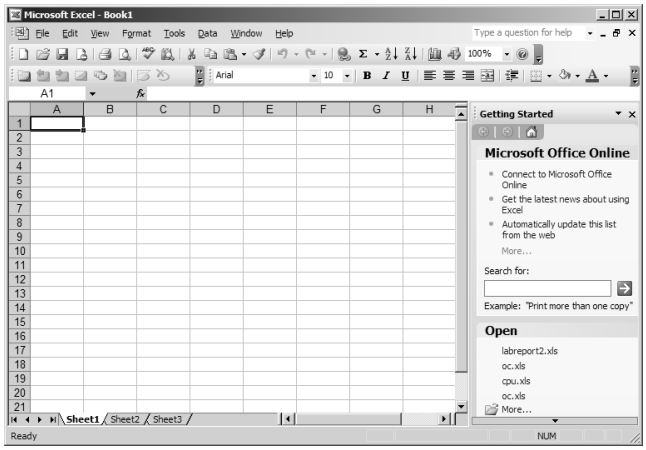
8.1 Visicalc

The concept of a spreadsheet was developed from the technique known as “running the numbers” as used at the Harvard Business School. Dan Bricklin developed the idea of using a computer by watching his lecturer use a table to record the results of financial calculations. If a mistake occurred somewhere in the process, the lecturer had to rub out any subsequent results, correct the error and recalculate all the entries again. Dan Bricklin got the idea that a computer could be used to automate this process and the results of any calculations could be displayed visually in a table. Together with Bob Frankston, a capable programmer, they created a program which they called Visicalc (standing for visible calculator).

The first copy was sold in October 1979, and it was only available for the Apple II. This application was so useful that people in middle management throughout the U.S.A. felt they had to have it. These people had comfortable jobs with disposable income, so they could afford the expense of a personal computer. Visicalc was so useful that people were willing to buy an Apple II computer just to run that one program. Apple's sales increased massively, driving the personal computer industry into the public eye, and making Apple's founders, Steve Jobs and Steve Wozniak, into household names.



Later, the rights to Visicalc were bought by Lotus, who produced Lotus 1-2-3 for the IBM PC, based on the Apple product. Other spreadsheets like Wings, ClarisWorks, OpenOffice Calc and Gnumeric followed. Today, the dominant spreadsheet is Microsoft Excel. It is widely used in both commercial and home environments. However, some of the statistical functions in Excel do not work correctly, so for serious academic research a dedicated statistics package or an alternative spreadsheet such as Gnumeric is recommended.

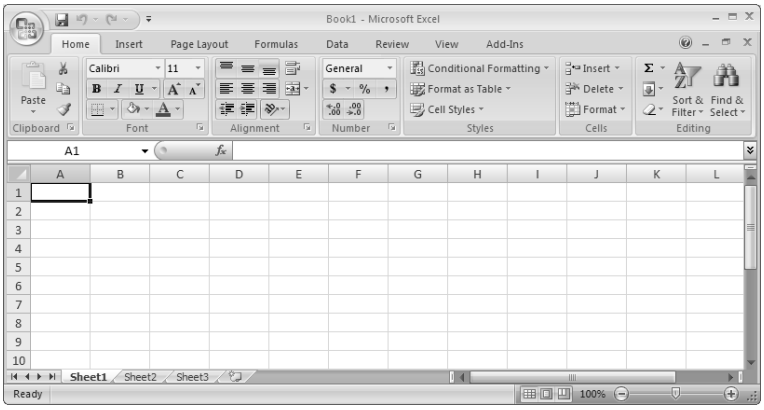


8.2 Introduction

A spreadsheet is an application that allows a user to enter data into a table. Once the data is entered, calculations can be performed using formulae, the data can be manipulated by various functions and the data can be formatted for presentation. Spreadsheets are commonly used by people working with tables of data.

8.2.1 Menus and Toolbars

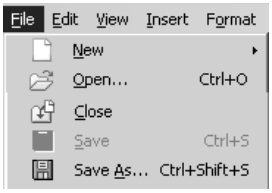
Different versions of Excel will have slightly different user interfaces (especially between the Office 2003 and Office 2007 versions), but the underlying principles remain the same. When we start a new Excel document, we are presented with a blank spreadsheet.



At the top of the window we can see a number of Menus and Toolbars. We use these to perform actions that apply to our documents. There are major differences between the way Office 2003 and Office 2007 use Menus and Toolbars.

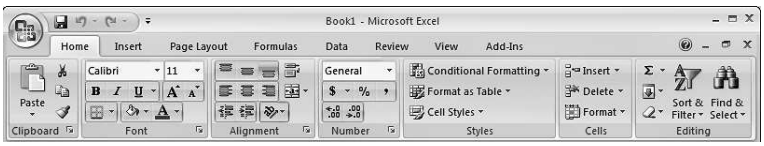
Office 2003

The menus work the same as with most other applications. When we click on a menu, a drop-down list appears that presents us with a series of options relevant to the topic of the menu. For example, a typical File menu appears as:

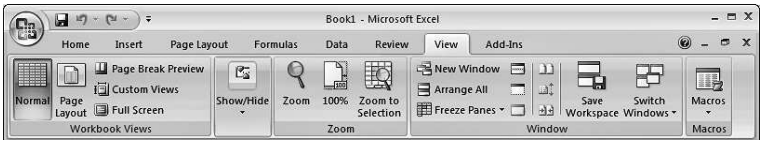


Office 2007 & Office 2010

The idea behind the menus has changed from previous versions. The main menus are now presented as a tabbed Toolbar. In other words, the items are no longer chosen by picking from a list, but rather they are presented as a Toolbar (this new toolbar is called a "'ribbon"' by Microsoft). For example, the Home menu results in the Home tab being displayed:



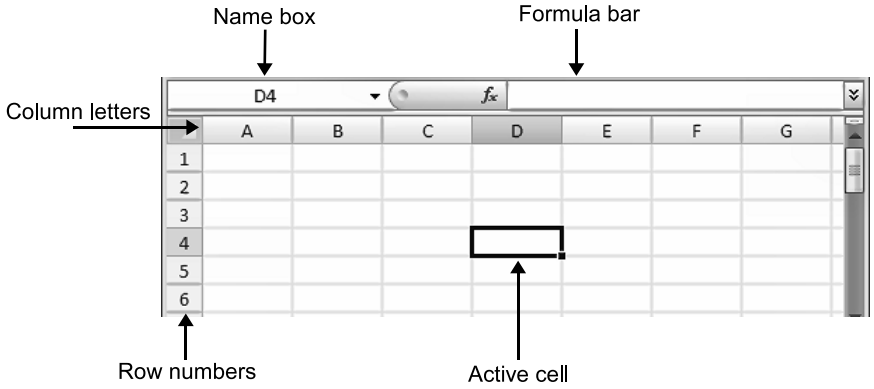
while the View menu results in the View tab being displayed.



If we hover the mouse over any of the Toolbar buttons then a pop-up box will appear explaining what the button does. In this course, we will be using Office 2010.

8.3 Adding data

Each position in the spreadsheet is known as a *cell*. Each cell can contain text, numbers, or a formula. The cell can also be formatted for a nicer appearance (borders, shading, colour etc.), but changing the appearance does not change the data stored in the cell.

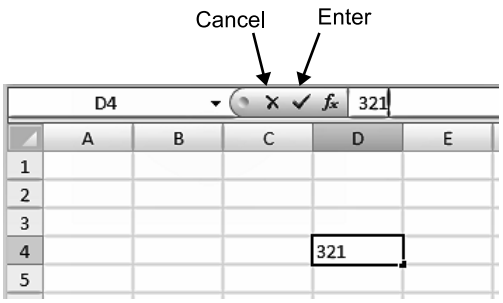


- Each column is identified by a letter (i.e. A, B, C. . . .).
- Each row is identified by a number (i.e. 1, 2, 3, . . .).
- The currently selected cell is known as the *active cell*, and it will be highlighted.
- The name box shows the name of the active cell. An individual cell in the spreadsheet is referenced using the combination of column letter followed by row number (e.g. A3, B7, D8). These references are needed in order to use formulae, which is where the real power of a spreadsheet lies.
- The formula bar will display the current contents of the active cell.

8.3.1 Entering data

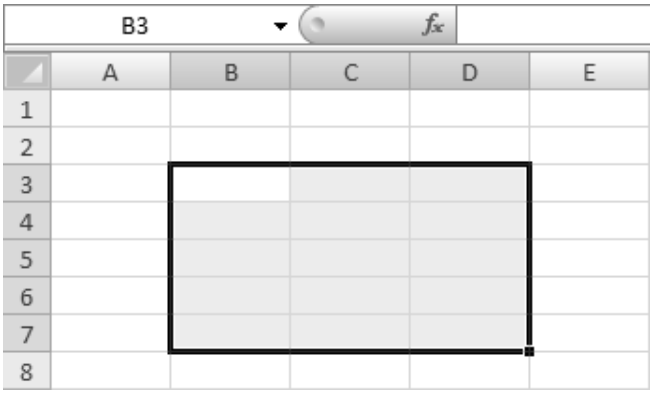
To enter information into any of the cells, simply click on the cell to make it the active cell (i.e. select the cell by clicking on it). Once the cell is active, any text that is entered will appear in the formula bar at the top of the spreadsheet. The data is not added to the spreadsheet until the Enter key is pressed, or until the Enter icon is clicked. If we

change our mind about entering data and wish to abort, we can click on the Cancel icon and the data will not be added to the spreadsheet.



8.3.2 Selecting a range of cells

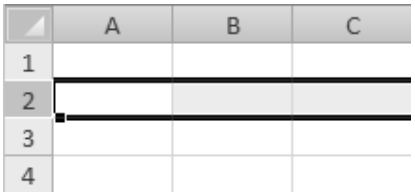
We can select a range of cells by clicking the mouse and dragging it across the spreadsheet. As you drag the mouse, we will see a number of cells are highlighted. In the following diagram, the active cell is B3, and the range of cells from B3 to D7 is selected.



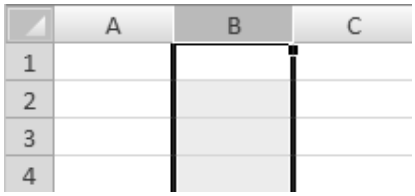
8.3.3 Selecting an entire row or column

To select an entire row, click on the number for that row. To select multiple rows, click on the number and drag the mouse across the rows we wish to select.

To select an entire column, click on the letter for that column. To select multiple columns, click on the letter and drag the mouse across the columns we wish to select.

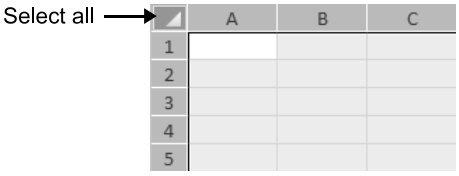


Selecting a row



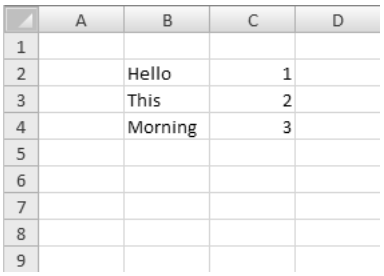
Selecting a column

To select the entire spreadsheet, click on the square between the row and column headers in the top left corner of the spreadsheet.

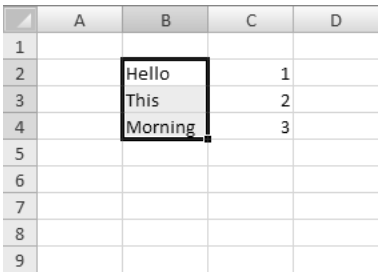


8.3.4 Copying and pasting

We can cut, copy and paste cells using the clipboard. In order to copy some spreadsheet cells, we should first highlight the cells we wish to copy.

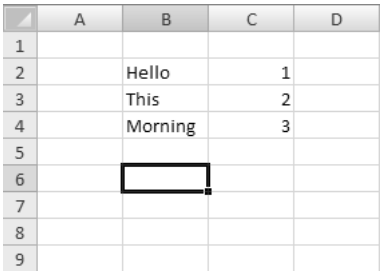


Original spreadsheet

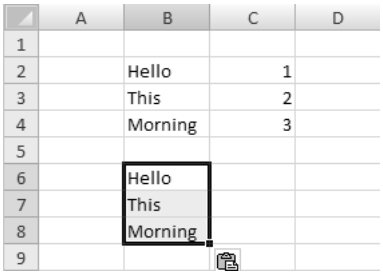


Highlight cells

Use the Copy icon from the *Clipboard* section of the Home tab, or use the keyboard shortcut Ctrl + C to copy the selected cells. A dashed line will appear around the cells that have been copied to the clipboard. To paste the selected cells, click on the cell that we want to be in the top left corner of the destination cells, then choose to paste using the Paste icon from the *Clipboard* section of the Home tab, or use the keyboard shortcut Ctrl + V.

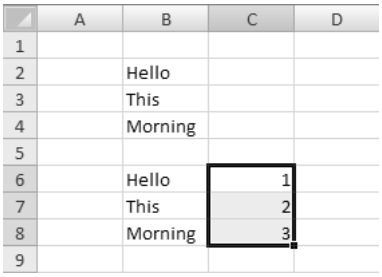


Destination cell



Completed paste

If we want to move the cells from one location to another, then we should use the Cut icon from the *Clipboard* section of the Home tab, or use the keyboard shortcut Ctrl + X. Use a similar process to the one described for copying.

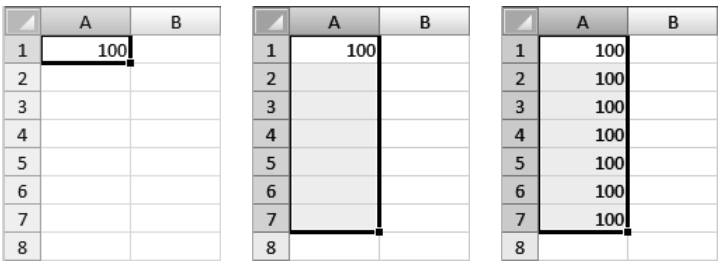


	A	B	C	D
1				
2		Hello		
3		This		
4		Morning		
5				
6		Hello	1	
7		This	2	
8		Morning	3	
9				

8.3.5 Filling data

Excel has a useful feature called *fill*. Instead of typing data into a series of cells, we can select the first cell and tell the spreadsheet to automatically fill the other cells with the contents of the first one. To fill a simple value, use the Fill icon from the *Editing* section of the Home tab, or use one of the following keyboard shortcuts:

- Fill down — Ctrl + D
- Fill right — Ctrl + R
- Fill up — Ctrl + U
- Fill left — Ctrl + L



	A	B
1	100	
2		
3		
4		
5		
6		
7		
8		

Original cell

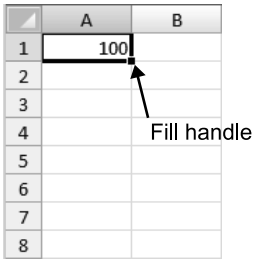
	A	B
1	100	
2		
3		
4		
5		
6		
7		
8		

Selected cells

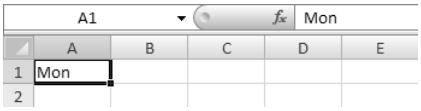
	A	B
1	100	
2	100	
3	100	
4	100	
5	100	
6	100	
7	100	
8		

Fill down

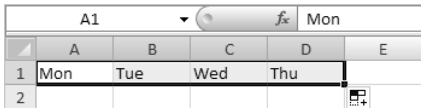
Alternatively, we can use the *fill handle*. If we click on the fill handle and drag the mouse to highlight a series of cells, the spreadsheet will automatically fill those cells with the contents of the first cell.



The fill handle can be used to automatically fill in data according to a predefined list. If the first item is recognised as part of the list, then Excel can automatically fill in the cells by cycling through the list. The predefined lists include days of the week and months.

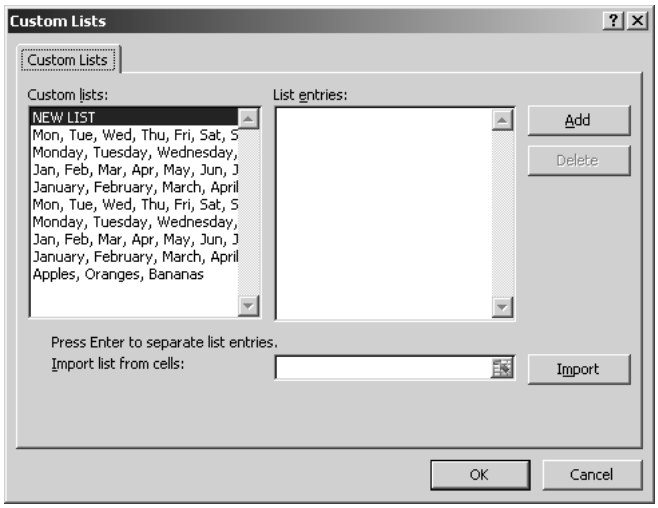


Original cell



Fill right with fill handle

We can customize the lists that Excel can recognise, so that it can automatically fill in a commonly used sets of data. To customize the list, use the File tab and click on Options. Click on Advanced on the left side, scroll down to the General section and click on the Edit Custom Lists button. The following dialog box will appear:



We can add our own custom list. In the dialog box shown above, the custom list {Apples, Oranges, Bananas} has been added.

8.3.6 Insert/delete rows and columns

As we are entering data into the spreadsheet, it is not uncommon to realise that we need an additional row or column. We can easily insert a new row or column so that we can add data in the correct place.

For example, in the following spreadsheet, we have entered the name and age of a number of people (perhaps members of a club). The names are currently listed in alphabetical order. If we have new data for another person (e.g. Emma, age 30) that we want to add into the middle of the existing data, then we need to insert a new row.

	A	B	C
1	Name	Age	
2	Ann	23	
3	David	17	
4	Greg	35	
5	Susan	54	
6			

To insert a new row, select the row that we want to move down. The new row will appear at this location and the remaining rows will be shifted down to make room.

	A	B	C
1	Name	Age	
2	Ann	23	
3	David	17	
4	Greg	35	
5	Susan	54	
6			

Original data

	A	B	C
1	Name	Age	
2	Ann	23	
3	David	17	
4	Greg	35	
5	Susan	54	
6			

Selecting the row where data is inserted

Once the row is selected, choose to Insert from the *Cells* section of the Home tab, or Insert → Insert Sheet Rows.

	A	B	C
1	Name	Age	
2	Ann	23	
3	David	17	
4			
5	Greg	35	
6	Susan	54	

Inserting a row

	A	B	C
1	Name	Age	
2	Ann	23	
3	David	17	
4	Emma	30	
5	Greg	35	
6	Susan	54	

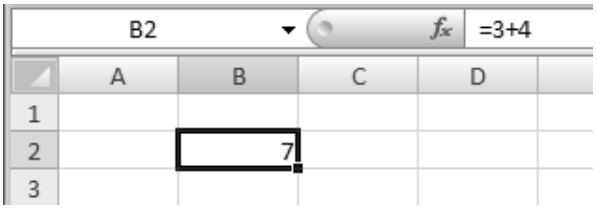
Entering the new data

Once a new row has been inserted into the spreadsheet, we can enter the data into the space available.

8.4 Formula

The real power of a spreadsheet is the ability to calculate the value to be used in one cell based on the values contained in other cells. To calculate a value, we need to use a formula. All formulae must start with the = sign.

We can use a formula to perform a calculation in the same way that we would use a calculator (e.g. to calculate to result of (`3 + 4`). This formula would be entered into a cell as `=3+4`. The result of the calculation (`7`), would be displayed on the page in the cell that contained the formula. Note that the cell *contains the formula*, but it *displays the result* of the formula.



Cell B2 contains the formula `=3+4`, but the value displayed is the result `7`

8.4.1 Relative references

Formulae may contain references to the contents of other cells. For example, in the following spreadsheet, we calculate the sum of two values by referring to the cells that

contain the data.

F9

	A	B	C	D
1	Hours Worked			
2	Name	Mon	Tue	Total
3	Alice	7	2	
4	Bob	3	4	
5	Cath	4	5	
6	David	6	2	

Employee hours worked

D3 =B3+C3

	A	B	C	D
1	Hours Worked			
2	Name	Mon	Tue	Total
3	Alice	7	2	9
4	Bob	3	4	
5	Cath	4	5	
6	David	6	2	

Calculating the total hours

Cell D3 contains the formula `=B3 + C3`. The formula effectively says "'take the value stored in cell B3 and add it to the value stored in C3, and the result of that calculation will be displayed in this cell"'.

It would be possible to type similar formulae into cells D4, D5 and D6 as follows:

- D4 contains formula `=B4 + C4`
- D5 contains formula `=B5 + C5`
- D6 contains formula `=B6 + C6`

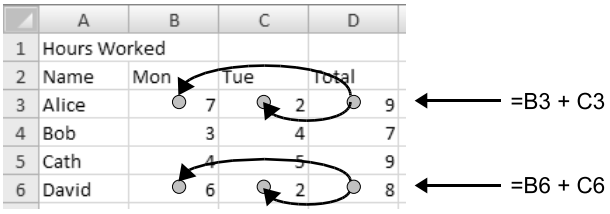
However, this process would quickly become tedious, especially if we had many rows of data. Thankfully, we can use *fill* to fill the formula into the adjacent cells.

D6 =B6+C6

	A	B	C	D	E
1	Hours Worked				
2	Name	Mon	Tue	Total	
3	Alice	7	2	9	
4	Bob	3	4	7	
5	Cath	4	5	9	
6	David	6	2	8	
7					

The references used in the formula are automatically changed when we fill.

Whenever we copy (or fill) a cell containing a formula to a different location, the formula is automatically changed to refer to different cells. These kind of cell references are known as relative references because the cells that are referred to in the formula are relative to the location of the formula in the spreadsheet. For example the cell we are referring to may be 2 columns to the left, and 1 row above the current cell. Wherever the formula is moved to, it would refer to the cell which is 2 columns left of the current cell and 1 row above it.



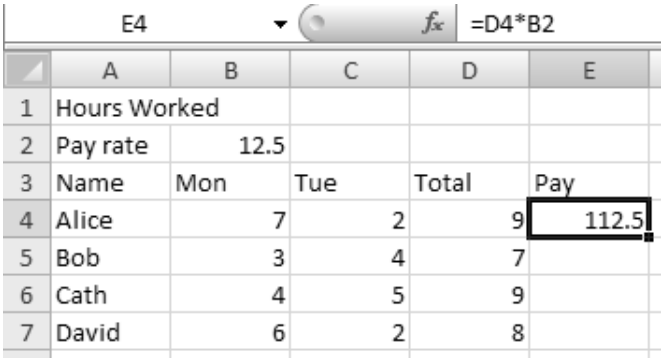
The cell references are relative to the location of the formula

In the example above, the formula in cell D3 is `=B3 + C3`. Because these are relative references, the spreadsheet interprets the formula as "'add the contents of the cell 2 to

the left of this one to the cell 1 to the left of this one"'. When the formula is moved to cell D6 the formula still says "'add the contents of the cell 2 to the left of this one to the cell 1 to the left of this one"', but because the new formula is in cell D6, the cells referred to by the formula are `=B6 + C6`.

8.4.2 Absolute references

Sometimes we want to refer to the same cell, no matter where the formula that refers to that cell is moved. For example, we might have a cell that contains the current hourly pay rate which we will want to refer to when we are calculating the pay for a large number of different employees.

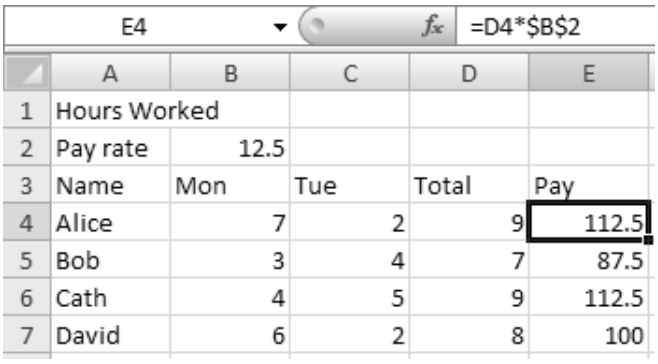


	A	B	C	D	E
1	Hours Worked				
2	Pay rate	12.5			
3	Name	Mon	Tue	Total	Pay
4	Alice	7	2	9	112.5
5	Bob	3	4	7	
6	Cath	4	5	9	
7	David	6	2	8	

The pay rate is always stored in cell B2

If we *fill* this formula to the cells below, then both the reference to the pay rate (B2) and the reference to the total hours (D4) will be changed. This will give the incorrect results. We note that we want to change the reference to the total hours when we move the formula, but the reference to the pay rate should remain fixed. We need a way to distinguish between these two different kinds of references.

A reference that should remain fixed is called an *absolute reference*. Excel uses the dollar sign immediately before the row or column (or both) that should remain fixed. If an absolute reference is used, and the formula is copied to a new location, then the absolute reference will not be changed. It will still refer to the same cell, regardless of where the formula is copied to.



	A	B	C	D	E
1	Hours Worked				
2	Pay rate	12.5			
3	Name	Mon	Tue	Total	Pay
4	Alice	7	2	9	112.5
5	Bob	3	4	7	87.5
6	Cath	4	5	9	112.5
7	David	6	2	8	100

The reference to the pay rate should be an absolute reference

In the example shown above, the formula contains both relative and absolute references. The formula can safely be filled to the selected cells below. The relative references will be automatically changed, but the absolute references will remain the same. The new formulae will be:

- E4 contains formula `=D4 * $B$2`
- E5 contains formula `=D5 * $B$2`
- E6 contains formula `=D6 * $B$2`

8.4.3 Good spreadsheet design

The most important design consideration about a spreadsheet is to make sure it calculates correct results. One of the most common mistakes that people make when they design the spreadsheet is to put the same information in more than one place. This initially works fine, but if the information is changed in only one place at a later stage, then the spreadsheet will be inconsistent and it will calculate the wrong results.

For example, in the previously discussed spreadsheet, we could have hard-coded the pay rate by putting the pay rate directly into the formula (e.g `=D4 * 12.5`). Although this formula would have worked, if the pay rate is changed later by altering the contents of cell B3, the spreadsheet would become inconsistent and the wrong results would be calculated.

E4 f_x =D4*12.5

	A	B	C	D	E
1	Hours Worked				
2	Pay rate	12.5			
3	Name	Mon	Tue	Total	Pay
4	Alice	7	2	9	112.5
5	Bob	3	4	7	87.5
6	Cath	4	5	9	112.5
7	David	6	2	8	100

Using numbers instead of cell references in formulae can lead to inconsistencies.

Although it can be time consuming to enter all the formulae into the spreadsheet, we can often reuse the same spreadsheet. For example, a spreadsheet used to calculate student's grades should be able to be reused each year. For this reason, we must try to construct spreadsheets so that the work required to recalculate results based upon a new set of data is kept to a minimum.

Example

The following spreadsheet might be used by a teacher to calculate the final grade for a student. In this example spreadsheet, there are 2 different assignments. The first assignment is marked out of 10. The second assignment is marked out of 7. The teacher wants each assignment to contribute equally to the final mark.

	A	B	C	D
1	Assignment marks			
2	Name	A1	A2	Final
3	Homer	1	2	
4	Marge	5	6	
5	Bart	3	2	
6	Lisa	10	7	
7	Maggie	0	1	

First, we could convert each mark to a percentage, then we could take the average of the two percentages.

We could calculate the final mark as follows:

- Convert the first mark to a percentage using (first mark / 10 * 100)
- Convert the second mark to a percentage using (second mark / 7 * 100)
- Calculate the average of the two percentages using (first percentage + second percentage) / 2

	A	B	C	D	E	F
1	Assignment marks					
2	Name	A1	A2	A1 %	A2 %	Final
3	Homer	1	2	10	28.57143	19.28571
4	Marge	5	6	50	85.71429	67.85714
5	Bart	3	2	30	28.57143	29.28571
6	Lisa	10	7	100	100	100
7	Maggie	0	1	0	14.28571	7.142857

The formulae used are as follows:

- D3 contains `=B3 / 10 * 100`
- E3 contains `=C3 / 7 * 100`
- F3 contains `=(D3 + E3) / 2`

Although these formulae work correctly now, it makes it difficult to reuse the spreadsheet without a lot of work. If the teacher decided that they wanted to use the same spreadsheet with assignments that were marked out of 20, then they would have to rewrite all the formulae.

Instead, we could record what each assignment was marked out of (i.e. the maximum mark that a student could obtain) and refer to the cells containing the information. This means that we could reuse the spreadsheet simply by changing the contents of those cells.

	A	B	C	D	E	F
1	Assignment marks					
2	Name	A1	A2	A1 %	A2 %	Final
3		10	7	100	100	100
4	Homer	1	2	10	28.57143	19.28571
5	Marge	5	6	50	85.71429	67.85714
6	Bart	3	2	30	28.57143	29.28571
7	Lisa	10	7	100	100	100
8	Maggie	0	1	0	14.28571	7.142857

The new formulae used are as follows:

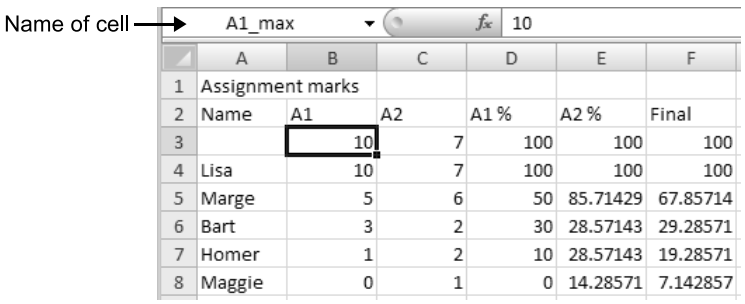
- D4 contains `=B4 / $B$3 * $D$3`
- E4 contains `=C4 / $C$3 * $E$3`

When we are creating a spreadsheet, it is best to have each piece of information stored only in one single cell. If that value is needed in another formula, then the formula should refer to the cell where the information is stored, rather than enter the actual value in a second place. It is much easier to keep the spreadsheet consistent and avoid mistakes using this approach.

8.4.4 Defining names

We can define a name for a cell, then we can use that name instead of the cell reference in the formulae that we write. This can sometimes make our formulae easier to understand.

Select the cell that we want to name and either click in the Name Box, or go to the Formulas tab and click on the Define Name icon, then type the name into the dialog box that appears.



	A	B	C	D	E	F
1	Assignment marks					
2	Name	A1	A2	A1 %	A2 %	Final
3		10	7	100	100	100
4	Lisa	10	7	100	100	100
5	Marge	5	6	50	85.71429	67.85714
6	Bart	3	2	30	28.57143	29.28571
7	Homer	1	2	10	28.57143	19.28571
8	Maggie	0	1	0	14.28571	7.142857

Once the cell is named, we can use the name in our formulae. For example, instead of referring to cell `B3` in the spreadsheet shown above, we can now refer to the cell `A1_max`. The formula in cell D4 could either be written as:

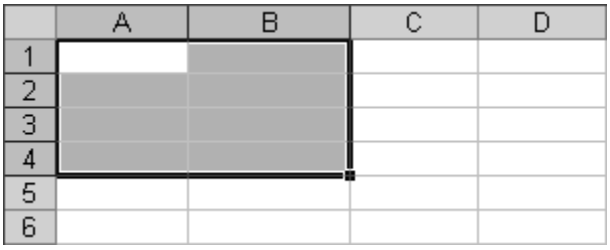
- `= B4 / $B$3 * $D$3` ;or
- `= B6 / A1_max * $D$3`

8.5 Functions

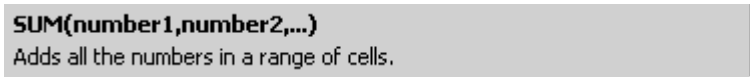
Most spreadsheets provide a large range of functions that help the user to create effective formulae. All functions use the same general structure.

```
FUNCTION-NAME( parameter1, parameter2, ..., parameterN )
```

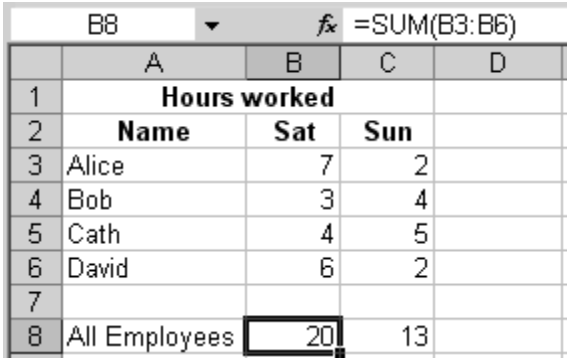
Some of the simple functions that we might use often are AVERAGE, MAX, MIN and SUM. These functions all operate on a range of cells. A range of cells is defined as a rectangular block of cells within the spreadsheet. In order to specify a cell range, we should give the top left cell, then a colon ':' and then the bottom right cell. In the example below, a range of cells has been selected. These cells can be referred to using `A1:B4`.



The SUM function is described in Excel as follows:



This description tells us that the SUM function is given a series of values and it calculates the sum of those values. Note that the description says that the function "Adds all the numbers in a range of cells". That means we can either refer to each individual cell in that range, or we can use the means of describing a range that we covered earlier. The following example shows how we would use the SUM function.



	A	B	C	D
1	**Hours worked**			
2	**Name**	**Sat**	**Sun**	
3	Alice	7	2	
4	Bob	3	4	
5	Cath	4	5	
6	David	6	2	
7				
8	All Employees	20	13	

The following spreadsheet is used to calculate the sum of the best 5 lab marks, chosen from a set of 6 labs. To find the best 5 marks, we can simply add up the marks from all 6 labs and subtract the smallest one. We could use the following functions to write the formula:

SUM Adds all the numbers in a range of cells
MIN Returns the smallest number in a set of values. Ignores logical values and text.

H2 =SUM(B2:G2) - MIN(B2:G2)

	A	B	C	D	E	F	G	H
1	**Name**	**Lab 01**	**Lab 02**	**Lab 03**	**Lab 04**	**Lab 05**	**Lab 06**	**Sum of 5 best**
2	Fred	3	3	6	2	6	7	25
3	Jane	3	9	9	8	4	3	33
4	Sally	9	2	3	4	6	2	24
5	Derek	10	4	9	1	4	7	34

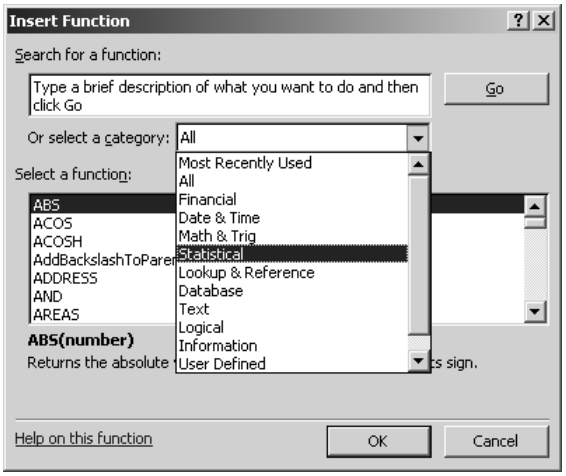
Functions allow a user to create formulae that involve complex calculations, without having to write all the calculations themselves. For example, to find the average mark scored for Lab 01 in the screenshot above, we could use any of the following three formulae:

- `=(B2 + B3 + B4 + B5) / 4`
- `=SUM(B2:B5) / COUNT(B2:B5)`
- `=AVERAGE(B2:B5)`

8.5.1 Inserting functions

The Formulas tab contains the tools used to insert various Excel formulae. The most frequently used section of this tab is the *Function Library* section.

All functions in Excel can be found by clicking the Insert Function icon. This will provide us with a dialog box containing all the functions. If we select a function, then we will be shown a brief description of what it does.



The functions supported by Excel are divided into different categories to make them easier to find. If we think there might be a function which does the job, have a look through the most likely category. All of the other buttons on the *Function Library* section

of the Formulas tab are used to access one or more of the function categories. These are:

Autosum The most common functions used to sum and count numbers (sum, average, count, min and max)

Financial Functions used predominantly in accounting and economics

Logical Functions used to make logical decisions

Text Functions used to manipulate text

Date and Time Functions used to work with times and dates

Lookup and Reference Functions used to look up values in tables

Math and Trig Traditional functions related to mathematics and trigonometry

Statistical Statistics functions

Engineering Functions used in engineering and technology

Information Functions that allow us to find out information about the type of value in a spreadsheet cell

8.5.2 Common mathematical and statistical functions

Statistical calculations are commonly required by the users of spreadsheet software, so it is no surprise that a large range of formulae have been provided as functions. The statistical category contains many of the common functions we will need (including minimum, maximum, quartiles, averages, and standard deviations). The most common functions that we use include:

- max, min, average, median, mode, sum
- stdev, quartile, correl, ttest, rsq

8.5.3 Counting functions

These functions are used to count cells. We must use different functions depending upon what we need to count.

```
COUNT( Cell_range )
```

The `COUNT` function is used to count the number of cells within the range which contain a numeric value.

```
COUNTIF( Cell_range, Criteria )
```

The `COUNTIF` function counts the number of cells in the range which match the criteria. The criteria can be text, a number or a cell reference. There are other ways of specifying a criteria, but we will consider only simple matches. In other words, the function will count all the cells which contain the same information as that given in the criteria. To count all the cells that contain the word “Waiter”, we could use:

```
=COUNTIF( D3:D15, "Waiter" )
```

If the text “Waiter” were located in another cell, say A2 for example, then we could use:

```
=COUNTIF( D3:D15, $A$2 )
```

8.5.4 Conditional Functions

A conditional function is used when we want to put different values into a cell depending upon a particular condition.

```
IF( logical_condition, value_if_true, value_if_false )
```

If the `logical_condition` is true, then the second argument (`value_if_true`) will be displayed in the cell, otherwise the last argument (`value_if_false`) will be displayed in the cell.

The logical condition should be some kind of logical test. At the simplest level, these tests are:

equal to (=) For example: `B3 = B4`

greater than (>) For example: `B3 > B4`

less than (<) For example: `B3 < B4`

greater than or equal to (>=) For example: `B3 >= B4`

less than or equal to (<=) For example: `B3 <= B4`

not equal to (<>) For example: `B3 <> B4`

For example, in the following spreadsheet we use an IF function to determine if an employee is overworked or not. The formula in cell C2 is `=IF(B2>40, "Working too hard", "Working")`

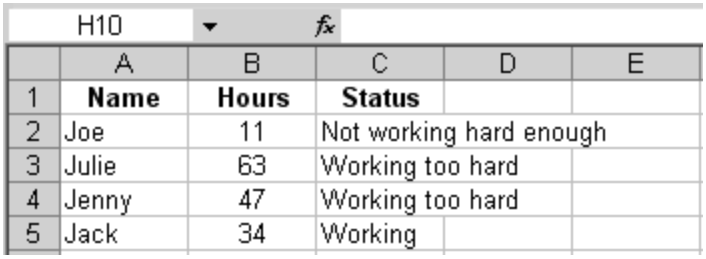
G9

	A	B	C	D
1	**Name**	**Hours**	**Status**	
2	Joe	11	Working	
3	Julie	63	Working too hard	
4	Jenny	47	Working too hard	
5	Jack	34	Working	

Note that we can use another function instead of a simple value where `value_if_true` or `value_if_false` appears in the IF function. In the following example, we have used the formula:

```
=IF(B2>40,
    "Working too hard",
    IF(B2<20, "Not working hard enough", "Working")
)
```

in cell C2.



	A	B	C	D	E
1	**Name**	**Hours**	**Status**		
2	Joe	11	Not working hard enough		
3	Julie	63	Working too hard		
4	Jenny	47	Working too hard		
5	Jack	34	Working		

Boolean operators

We can also use the standard boolean operators (AND, OR) to make a compound condition.

```
AND( logical_1, logical_2, ... )
```

The `AND` function returns true if all the arguments are true.

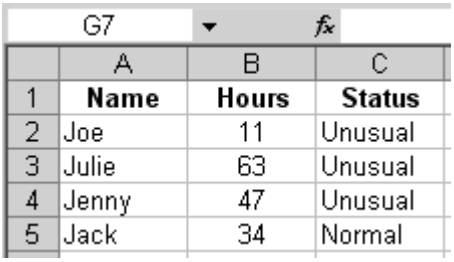
```
OR( logical_1, logical_2, ... )
```

The `OR` function returns true if any of the arguments are true.

In the following example, we use the formula:

```
=IF(OR(B2<20, B2>40), "Unusual", "Normal")
```

in cell C2 to determine if the person is spending a normal amount of time at work. We could alternatively write the formula as `=IF(AND(B2>=20, B2<=40), "Normal", "Unusual")` and it would produce the same result.



	A	B	C
1	**Name**	**Hours**	**Status**
2	Joe	11	Unusual
3	Julie	63	Unusual
4	Jenny	47	Unusual
5	Jack	34	Normal

8.5.5 Information functions

One useful function that returns some information about a cell is `ISBLANK( cell )`. This function is used to tell if a given cell contains any information, or if it is blank. The function will return a boolean value, that is, either `true` or `false`. For example:

```
=ISBLANK(A3)
```

will return `true` if cell A3 is empty. Since the function returns a boolean value, it can be used in a conditional function. For example:

```
=IF(ISBLANK(A3), "Cell is empty", "Cell contains information")
```

8.5.6 Lookup functions

These functions are designed to look up a value in a table and return a result from a particular row or column within the table. These lookup functions are extremely useful.

```
VLOOKUP( lookup_value, table_array, column_index_number, range_lookup )
```

This function is designed to look up a value in a defined table, and return an element of data from within the table. We will look at each of the arguments in turn.

`lookup_value`

The `lookup_value` is the value that the function tries to find in the table. It will look in the left-most column. When the appropriate table entry is located, the function returns a value from that row of the table. The value it returns is dependent on the `columnIndexNumber`.

`table_array`

The `table_array` is the range of cells that defines the table that we use when we look up values. The table should consist of at least two columns.

`column_index_number`

The `column_index_number` specifies the column number in the table from which the data will be returned. For example, a `column_index_number` of 1 will return a value from the left-most column of the table. A `column_index_number` of 2 will return the data from the second column of the table. Increasing the `column_index_number` will return the data from columns further to the right.

`range_lookup`

The `range_lookup` value is a boolean value. It must be either `TRUE` or `FALSE`. If the `range_lookup` value is `TRUE`, then it means that the table represents a range of values. The values that are listed in the left-most column of the table are treated as "boundary" values that define the start of a range. For example, the following table is used to describe age boundaries for movie tickets:

Age	Category
0	Free
2	Child
12	Adult
65	Senior Citizen

We can use this table to look up any age because we interpret the age values listed in the table as boundary values. If a person is at least 12 years old, and is not yet 65, then they pay Adult prices at the movies. Any person under the age of 2 is allowed in free of charge. A table that is to be interpreted this way (as representing a range of values) will require the `range_lookup` value to be `TRUE` in any `VLOOKUP` function that is used with the table.

If the `range_lookup` value is `FALSE`, then we expect the exact value we are looking up to appear in the table. For example, a table that contains student ID numbers should not be treated as a range of values, instead, we are looking for an exact match with the ID number we are looking up. In this case, the `range_lookup` value should be set to `FALSE`.

8.5.7 VLOOKUP Examples

The following example shows a spreadsheet that contains a `VLOOKUP` function. Consider the formula contained in cell D10. The formula is used to look up the job of the employee in order to find out the pay rate. Each of the arguments used in the `VLOOKUP` function are described below:

- `lookup_value`: The value that we want to look up is the job of the employee. That value is stored in cell B10. Since we want to fill this formula down and reuse it for other employees, the cell reference should be relative (in other words, the employee listed in row 11 would have their job listed in cell B11, so we want our formula to refer to the information in column B and have the row automatically adjusted as the formula is filled down).
- `table_array`: The table that contains the list of jobs and associated pay rates starts in cell A2 and ends in cell B6. This defines the table that we will use when we look up the pay rate for the job specified. Note that we do not include the titles of each column. The first row listed in the table is the first row that we might expect to find the information we are looking for. Since we always refer to the same table, regardless of where the `VLOOKUP` formula is located, the references used in the `table_array` should be absolute references.
- `column_index_number`: We always use the first column of the table to look up the value (in this case the job). Once the job has been located, we want to know the pay rate. In this example, the pay rate is located in the second column of the table that we use to look up the value. Since the information we want is in the second column, the `column_index_number` that we use is set to 2.
- `range_lookup`: We are looking up the exact word "Programmer" in the table. The table does not represent a range of values, rather each row in the table is independent from the others. In this case, since the table does not represent a range, we use the value `FALSE` for the `range_lookup`.

D10 f_x =VLOOKUP(B10,$A$2:$B$6,2,FALSE)

	A	B	C	D	E	F
1	**Job**	**Pay rate**				
2	Cleaner	12				
3	Programmer	60				
4	Doctor	100				
5	Manager	30				
6	Teacher	10				
7						
8						
9	**Employees**	**Job**	**Hours**	**Pay rate**	**Wages**	
10	Isabella	Programmer	15	60	900	

In this second example, the spreadsheet shown below contains a `VLOOKUP` function in cell C12. The `VLOOKUP` function is used to look up the Saffir-Simpson Hurricane Category based on the wind speed recorded on a given date. Each of the arguments used in the `VLOOKUP` function are described below:

- `lookup_value`: The value that we want to look up is the recorded wind speed (in miles per hour) for a given date. That value is stored in cell B12. Since we want

to fill this formula down and reuse it for other dates, the cell reference should be relative.

- `table_array`: The table that contains the Saffir-Simpson Hurricane Scale starts in cell A3 and ends in cell B8. This defines the table that we will use when we look up the wind speed. Note that we do not include the titles of each column. The first row listed in the table is the first row that we might expect to find the information we are looking for. Since we always refer to the same table, regardless of where the `VLOOKUP` formula is located, the references used in the `table_array` should be absolute references.
- `column_index_number`: We always use the first column of the table to look up the value (in this case the wind speed). Once the appropriate row has been located, we want to know the corresponding Hurricane Category. In this example, the wind speed is greater than the largest value listed, so we use the last row in the table to look up the value. Since the information we want is in the second column, the `column_index_number` that we use is set to 2.
- `range_lookup`: In the Saffir-Simpson Hurricane scale, the same hurricane category is used to describe a hurricane that blows with a range of different windspeeds. For example, any hurricane that has a wind speed between 96 and 110 inclusive is a category 2 hurricane. Once the wind speed reaches 111 mph, the hurricane is treated as belonging to the next highest category (category 3).
  The table in this case is used to represent a range of different values, so we set the `range_lookup` value to be `TRUE`. The numbers that appear in the left-hand column of the table are the boundary values that define the different categories. If the number that we are looking up is at least equal to the number in the current row, but not as high as the number in the following row, then we know that it belongs to the current category. Since the number we are looking up is 177 mph, that is at least as high as 155, so it is considered to be in the highest category, so the function returns hurricane category 5.

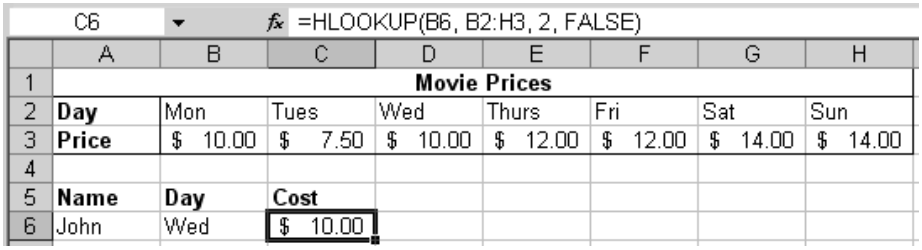
C12 =VLOOKUP(B12,$A$3:$B$8,2,TRUE)

	A	B	C	D
1	**Saffir-Simpson Hurricane Scale**			
2	**Wind Speed (mph)**	**Hurricane Category**		
3	0	0		
4	74	1		
5	96	2		
6	111	3		
7	131	4		
8	155	5		
9				
10				
11	**Date**	**Recorded Wind(mph)**	**Category**	
12	14-Nov-59	177	5	
13	12-Jan-53	48	0	
14	17-Jan-68	89	1	
15	13-Sep-66	99	2	
16	2-Oct-26	147	4	

```
HLOOKUP( lookup_value, table_array, row_index_number, range_lookup )
```

This function is designed to look up a value in a defined table, and return an element of data from within the table. The only difference between this lookup function and `VLOOKUP` is that this function is used to look up values in a horizontal table, whereas `VLOOKUP` is used for a vertical table.

For example, the following spreadsheet is used to figure out the price of movie tickets depending on the day of the week. Because the days of the week are listed horizontally, we use an `HLOOKUP` function.



C6 =HLOOKUP(B6, B2:H3, 2, FALSE)

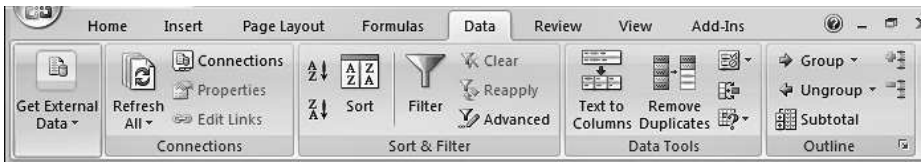
	A	B	C	D	E	F	G	H
1			**Movie Prices**					
2	**Day**	Mon	Tues	Wed	Thurs	Fri	Sat	Sun
3	**Price**	$ 10.00	$ 7.50	$ 10.00	$ 12.00	$ 12.00	$ 14.00	$ 14.00
4								
5	**Name**	**Day**	**Cost**					
6	John	Wed	$ 10.00					

The formula located in C6 is used to look up the day of the week. We will consider each of the arguments used in the `HLOOKUP` function:

- `lookup_value`: The value that we want to look up is the day of the week. That value is stored in cell B6.
- `table_array`: The table that contains the movie prices for each day starts in cell B2 and ends in cell H3. Note that we do not include the titles of each row. Since we always refer to the same table, regardless of where the `HLOOKUP` formula is located, the references used in the `table_array` should be absolute references.
- `row_index_number`: We always use the first row of the table to look up the value (in this case the day). Once the day has been located, we want to know the cost of tickets on that day. In this example, the ticket price is located in the second row of the table that we use to look up the value. Since the information we want is in the second row, the `row_index_number` that we use is 2. Do not get confused between the numbers used to represent each row in the entire spreadsheet and the `row_index_number` used in an `HLOOKUP` formula. The `row_index_value` is relative to the location of the table. In other words, the first row in the table is considered to be row 1, the second is row 2 and so on, regardless of where the table is located in the spreadsheet.
- `range_lookup`: We are looking up the exact word “Wed” in the table. The table does not represent a range of values, rather each column in the table is independent from the others. In this case, since the table does not represent a range, we use the value `FALSE` for the `range_lookup`.

8.6 Sorting, filtering and removing duplicates

The data tab contains the tools that allow us to manipulate data by sorting, filtering and removing duplicates.

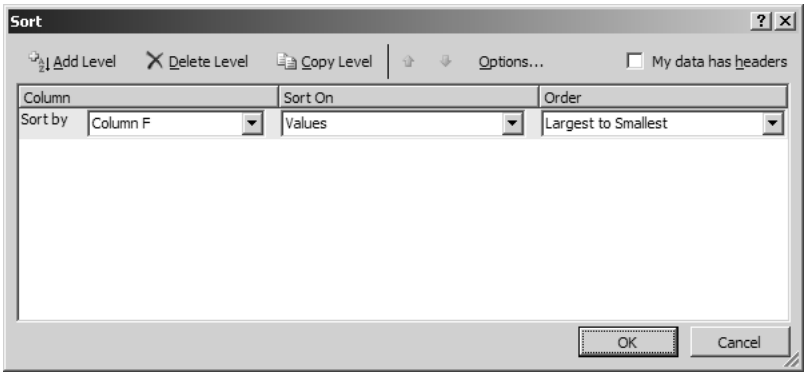


8.6.1 Sorting

Sorting data is a common operation when using spreadsheets. In order to sort data, first select all the data which is to be sorted. We can either:

- From the Home tab, select the Sort and Filter icon from the *Editing* section, and choose Custom Sort from the drop-down list; or
- From the Data tab, select the Sort icon from the *Sort and Filter* section

A dialog box will appear, allowing us to select which row or column should be used to sort the rows (to sort from left to right, click on the options button in the sort dialog box).

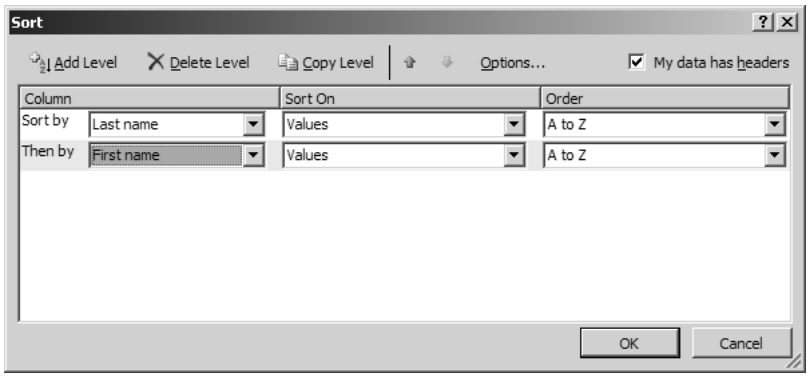


We can add further levels to the sorting criteria in case the first element is equal. For example, if we wanted to sort the following names into order, based on last name and then first name, we first need to select the entire set of data.

	A	B
1	First name	Last name
2	John	Smith
3	Jane	Smith
4	Fred	Dagg
5	Fred	Fish
6	Bob	Smith
7	Gordon	Bennett
8	Joe	Bloggs

We could specify two levels of sorting, first the last name will be used, then the first name will be used. Note that the check box specifying “My data has headers” has been selected because we selected all the rows of the table including the names of the columns. The headers are shown in the column field of the sort dialog box (in

other words, it uses the names of our columns rather than just the letters column B and column A).



The spreadsheet data will be sorted as follows:

	A	B
1	First name	Last name
2	Gordon	Bennett
3	Joe	Bloggs
4	Fred	Dagg
5	Fred	Fish
6	Bob	Smith
7	Jane	Smith
8	John	Smith

Important warning: we need to make sure that we select *all* the data that we need to sort. If we only select only some of the columns then the data might end up mismatched.

8.6.2 Filtering

Filtering the data allows us to decide which of the rows we want to display. This can be a very useful feature. For example, given a list of people, we might only want to display the females. To filter some data, first select the entire set of data, then click on the Filter icon in the *Sort and Filter* section of the Data tab.

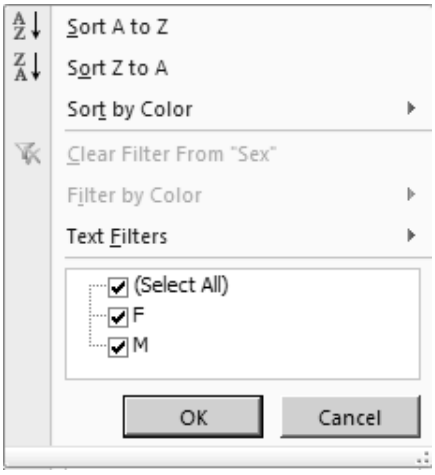
	A	B	C
1	First name	Last name	Sex
2	Gordon	Bennett	M
3	Joe	Bloggs	M
4	Fred	Dagg	M
5	Felicity	Fish	F
6	Bob	Smith	M
7	Jane	Smith	F
8	John	Smith	M

The original data

	A	B	C
1	First name	Last name	Sex
2	Gordon	Bennett	M
3	Joe	Bloggs	M
4	Fred	Dagg	M
5	Felicity	Fish	F
6	Bob	Smith	M
7	Jane	Smith	F
8	John	Smith	M

Selecting the data and choosing to filter

Once the data is ready to be filtered, we can click on the small icon that appears at the top of the columns that can be filtered. If we click on one of the filtering icons, a dialog box will appear that displays all the different values that appear in that column. Using this dialog box, we can select which values we want to display. Any row that contains information that we are filtering will be hidden from view.



If we decide that we will only display those rows that contain an "F" in the column entitled "Sex" then we effectively filter out all the males.

	A	B	C
1	First name	Last name	Sex
5	Felicity	Fish	F
7	Jane	Smith	F

8.6.3 Remove duplicates

Occasionally, we may wish to remove duplicate entries in a spreadsheet. In this case, simply select the data we wish to alter and click the Remove Duplicates icon in the *Data Tools* section of the Data tab.

8.7 Freezing, locking and hiding cells

8.7.1 Freezing cells

In large spreadsheets, it is often useful to have the row and/or column headings visible when we scroll through the data. We can *freeze* cells to ensure that they remain in place when the rest of the spreadsheet scrolls.

To freeze a row or column, go to the View tab, then click on the Freeze Panes icon and choose Freeze Top Row or Freeze First Column. If we want to freeze both rows and columns, then select the cell immediately below and to the right of the

row and columns that we want to freeze, then use the Freeze Panes icon to choose Freeze Panes

	A	B	C	D	E	F
1		Q1	Q2	Q3	Q4	Q5
2	John	1	4	2	1	9
3	Emma	1	2	3	5	1
4	Alice	9	5	5	2	2
5	Frank	6	4	9	8	6
6	Bob	7	4	5	2	4
7	Cath	7	9	5	6	4
8	Robin	9	7	8	9	2

Selecting Freeze Panes will freeze both the top row and first column

	A	D	E	F
1		Q3	Q4	Q5
3	Emma	3	5	1
4	Alice	5	2	2
5	Frank	9	8	6
6	Bob	5	2	4
7	Cath	5	6	4
8	Robin	8	9	2

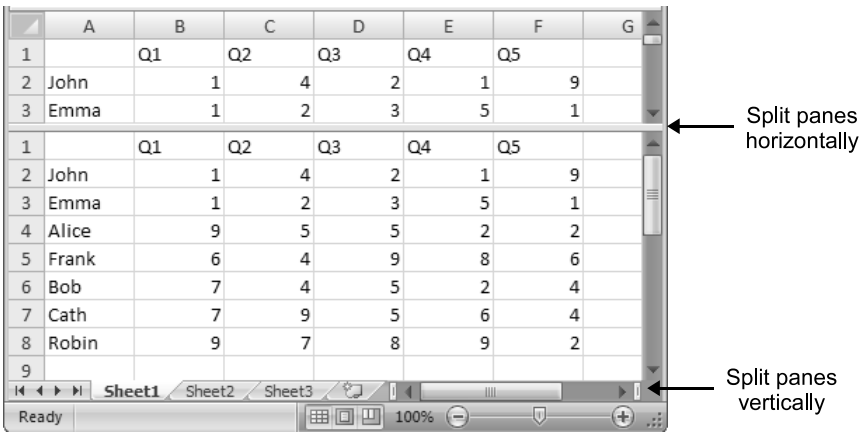
As the first row and column are locked, they remain fixed in place as the spreadsheet is scrolled. This spreadsheet has been scrolled so that columns D, E, F . . . and rows 3, 4, 5, . . . are visible.

We can unfreeze the panes by selecting Unfreeze Panes from the Freeze Panes icon.

8.7.2 Splitting panes

As an alternative to freezing rows and columns, we can split the spreadsheet view into multiple panes. If we hover the mouse over the small icons at the right of the horizontal scroll bar, or the top of the vertical scroll bar, the mouse pointer will change into a split icon.

Click and drag the icon to create a separate pane. We can scroll each pane independently. This effectively gives us multiple views of the same document.



	A	B	C	D	E	F	G
1		Q1	Q2	Q3	Q4	Q5	
2	John	1	4	2	1	9	
3	Emma	1	2	3	5	1	

1		Q1	Q2	Q3	Q4	Q5
2	John	1	4	2	1	9
3	Emma	1	2	3	5	1
4	Alice	9	5	5	2	2
5	Frank	6	4	9	8	6
6	Bob	7	4	5	2	4
7	Cath	7	9	5	6	4
8	Robin	9	7	8	9	2
9						

A window that has been split into two different panes.

Note: We cannot use split panes and frozen cells at the same time.

8.7.3 Hide and display cells

We can hide rows and columns that contain our "working", so that we only see the cells containing the information that is important. To hide a row or column, select the rows or columns that we want to hide.

	A	B	C	D	E	F
1		Q1	Q2	Q3	Q4	Q5
2	John	8	3	6	10	7
3	Emma	5	3	3	9	4
4	Alice	9	7	7	5	6
5	Frank	8	5	6	8	1
6	Bob	4	6	8	7	4
7	Cath	4	10	1	3	7
8	Robin	6	7	5	6	2

Once the rows or columns are selected, go to the Home tab and in the *Cells* section, choose Format → Hide and Unhide → Hide Rows or Hide Columns.

	A	E	F
1		Q4	Q5
2	John	1	2
3	Emma	1	7
4	Alice	9	9
5	Frank	6	2
6	Bob	7	5
7	Cath	9	5
8	Robin	3	9

To unhide the rows or columns, select the rows or columns either side of the ones hidden, then choose Format → Hide and Unhide → Unhide Rows or Unhide Columns.

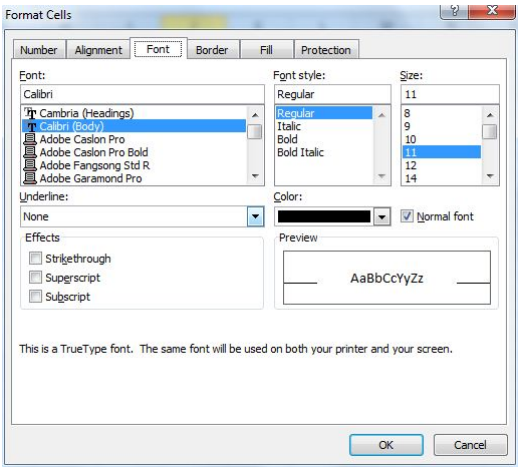
8.8 Cell Formatting

There are many tools available to change the appearance of the cells.

8.8.1 Font formatting

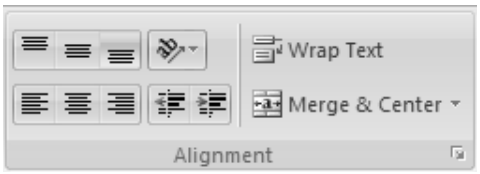
The tools for formatting text in Excel are similar to those used in word processing applications. We can apply multiple changes to the text one attribute at a time by clicking on the appropriate formatting icon in the toolbar, or we can change many attributes at the same time using the font formatting dialog box (e.g. we could make the text appear in Arial font typeface with 24 point size, bold, italic and underlined).

Choose the Home tab and click on the icon that displays the font dialog box in the *Font* section.



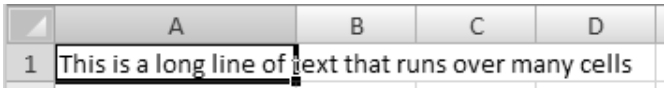
8.8.2 Alignment

The alignment section contains the tools to change the vertical and horizontal alignment of the contents of a cell, the orientation, whether text in the cell is wrapped or not, and the ability to merge cells.

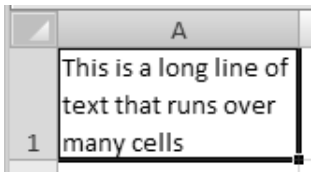


Wrap text in a cell

By default, text that is entered into a cell will extend into adjacent cells to the right. It often looks nicer to wrap the text to the size of the cell. Select the cell containing the text and click on the Wrap Text icon in the *Alignment* section of the Home tab.



Before wrapping the text



After wrapping the text

Merge and split cells

Merging cells is used to combine multiple cells into one new cell. This is commonly used for headings that run across multiple columns.

	A	B	C	D
1	Survey data			
2	Number	Q1	Q2	Q3
3	1	3	5	2
4	2	1	3	4

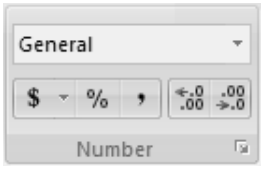
Before merging the cells

	A	B	C	D
1	Survey data			
2	Number	Q1	Q2	Q3
3	1	3	5	2
4	2	1	3	4

After merging the cells

8.8.3 Number formatting

Since the real power of spreadsheets is working with numbers, it is not surprising that there are many formatting options that define how numbers are displayed.



The number section of the Home tab.

We can select the kind of number that is represented by clicking on the Number Format drop-down list. This allows us to specify that the number is actually a percentage, currency, date, time, or should be displayed in scientific notation etc.

We can also adjust the number of decimal places used to display the number by clicking on the Increase Decimal or Decrease Decimal buttons.

8.8.4 Cell Formatting

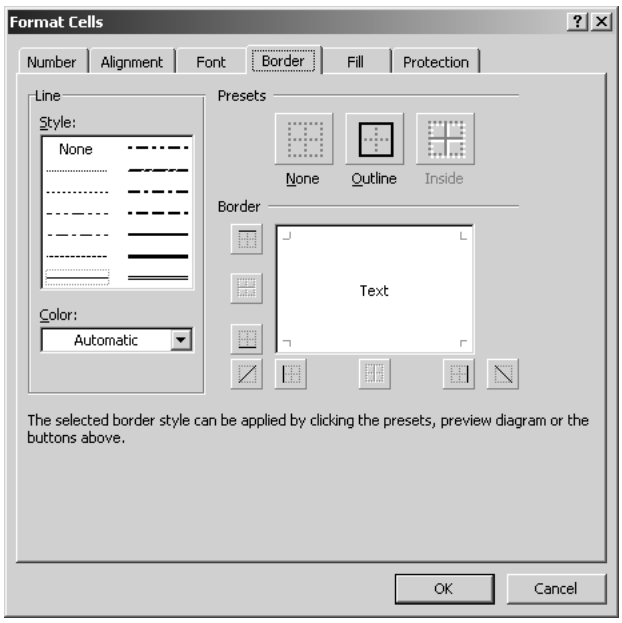
We can adjust the row height and column width by clicking on the Format icon in the *Cells* formatting section. We can also adjust the width of the columns and height of the rows by moving the mouse onto the dividing line between two cells in the row or column heading area.

Click and drag the mouse to change the size of the columns

The borders used around the cells can be altered by clicking on the Borders icon in the *Font* section of the Home tab.

The colour and texture (Shading) used to fill the cells can be altered by clicking on the Fill icon in the *Font* section of the Home tab.

All of the cell formatting details can be altered by accessing the Cell Formatting dialog box, which we can find by clicking on the Format icon in the *Cells* section of the Home tab, and choosing Fomat Cells from the drop-down menu.



8.8.5 Example

Formatting the cells correctly can make a dramatic difference to the appearance of the spreadsheet.

D3 =B3+C3

	A	B	C	D
1	Hours Worked			
2	Name	Mon	Tue	Total
3	Alice	7	2	9
4	Bob	3	4	
5	Cath	4	5	
6	David	6	2	

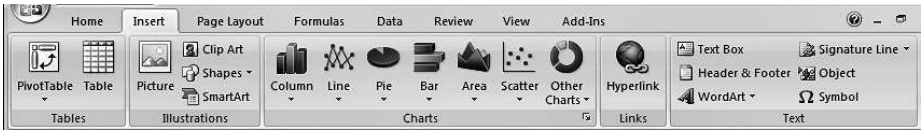
The hours worked spreadsheet before formatting the cells

	A	B	C	D	E
1	**Hours Worked**				
2					
3	*Pay rate*	*$12.50*			
4					
5	**Name**	**Mon**	**Tue**	**Total**	**Pay**
6	Alice	7	2	9	$112.50
7	Bob	3	4	7	$87.50
8	Cath	4	5	9	$112.50
9	David	6	2	8	$100.00
10					
11	Totals	20	13	33	**$412.50**

The hours worked spreadsheet after formatting

8.9 Charts

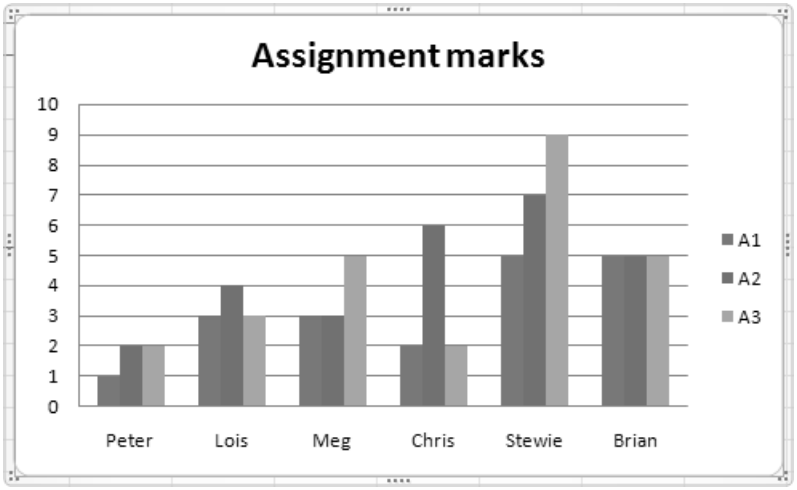
Use the Insert tab to create charts for our data.



To make a chart based upon the data in the spreadsheet, first we must select the data (including the headings for each row or column).

7	Student	A1	A2	A3
8	Peter	1	2	2
9	Lois	3	4	3
10	Meg	3	3	5
11	Chris	2	6	2
12	Stewie	5	7	9
13	Brian	5	5	5

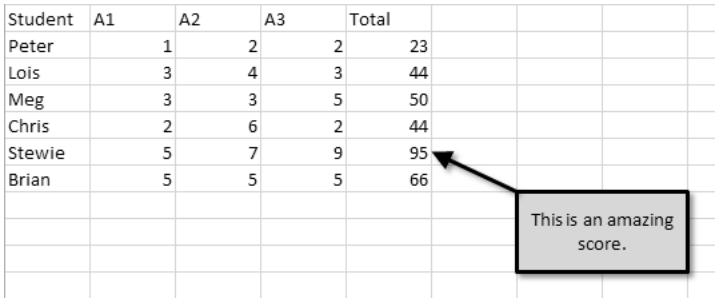
Go to the Insert tab and choose the chart type that we want from the *Chart* section.



The appearance of the chart can be altered significantly by choosing different options from the Chart Tools — Design, Chart Tools — Layout and Chart Tools — Format tabs.

8.10 Annotating data using the drawing tools

The drawing tools provided with Excel are similar to those provided with other Office products such as PowerPoint and Word. We can use these drawing tools to annotate our data with diagrams.

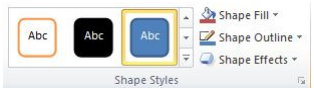


8.10.1 Shapes

Office 2010 (including Excel) has many pre-defined shapes that we can choose from. Once we select the shape that we want to draw, then clicking and dragging the mouse will draw the chosen shape on the spreadsheet. After the shape is drawn, we can alter various properties such as the outline and fill.

Shapes can be chosen from the Insert tab, by clicking on the Shapes button in the *Illustrations* section. To change the properties of the object, select the object (by clicking on it) and choose the Drawing Tools – Format tab. Alternatively, double-clicking the shape will automatically select the Drawing Tools – Format tab.

The "Shape Style" section of the tab allows us precise control over the appearance of the selected object.



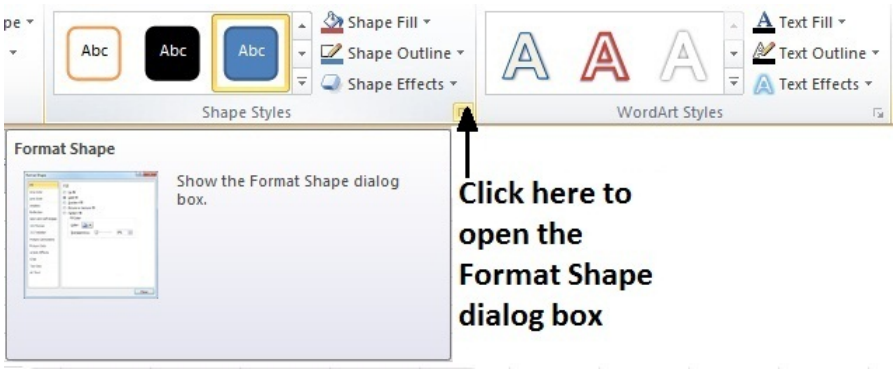
Each object (including any text boxes) floats in a different layer (imagine that each object is drawn onto a separate pane of glass and all of the panes are stacked up into a big pile). We can alter the order of these layers which in turn defines which objects appear in front of other objects.

To change the ordering of objects, choose Drawing Tools – Format then select the appropriate button from the *Arrange* section.

Objects can also be rotated or flipped (like a mirror image) using options from the menus described above. If we select a number of shapes, we have the option to Group the objects together (so that they are treated as a single object thereafter), or to Align the objects.

If we want to add text to a shape, simply start typing when that shape is selected. We can change the properties of the text by selecting it and using the normal menus to alter the appearance of the font.

If we want to alter the way the text is anchored to the shape, then use the Format Shape dialog box and select Text Box from the list of options.



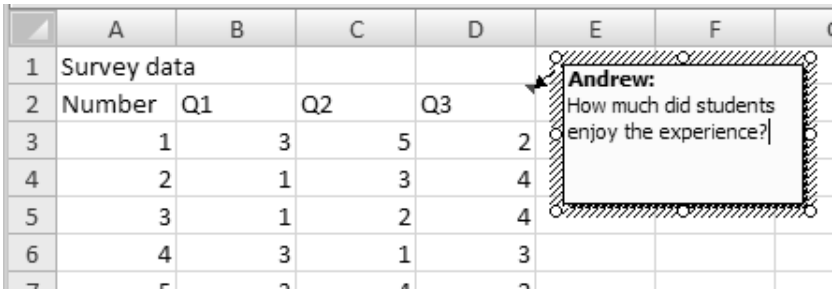
8.10.2 Grid

Excel allows us to align our drawing objects to the edges of the cells shown on the screen (these cells are described as the grid with respect to the drawing tools). When we create or move a drawing object, the edges of the object will always be aligned to one of the grid lines (i.e. the object will "jump" in size or location from one grid line to the next). If we want to align the drawing objects to the grid, then from the Drawing Tools tab, click on the Align icon in the *Arrange* section and choose Snap to Grid

Note that the Drawing Tools tab is only displayed when we have a drawing object selected.

8.11 Adding comments to cells

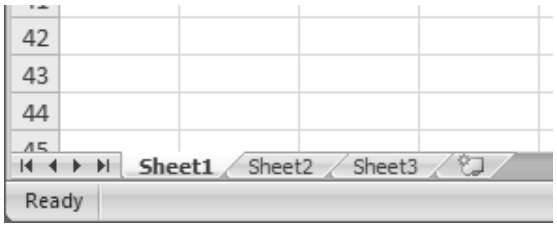
We have the ability to add a comment to a cell. This is especially useful if multiple people will be working on the same spreadsheet. With the cell selected, go to the Review tab and choose New Comment from the *Comments* section.



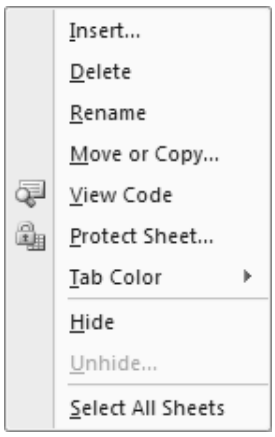
Once the comment is added, a small red triangle will appear in the top-right corner of the cell. Whenever the mouse is moved over that cell, the comment will appear. This feature is sometimes useful to separate the data from commentary about the data.

8.12 Multiple worksheets

A worksheet is like a single sheet of paper, consisting of all the rows and columns that we have been looking at so far. Excel allows us to have multiple worksheets in the same file. Excel calls the file consisting of all the worksheets a workbook.

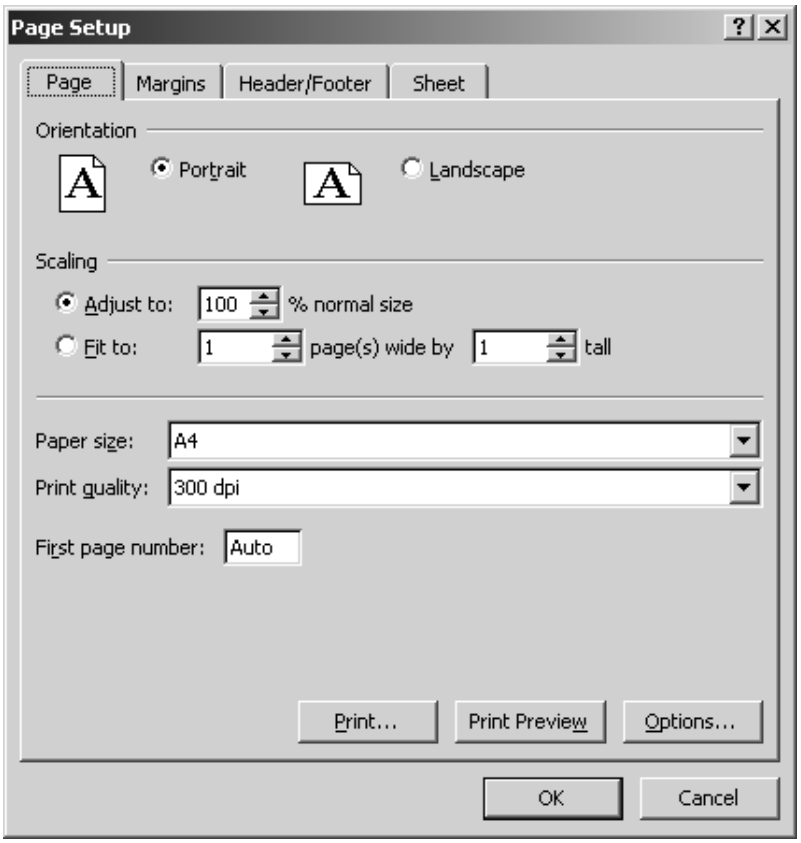


We can switch between the different worksheets by clicking on the name of the worksheet in the tabs along the bottom of the window. If we want to rename the worksheet, then we can right-click on the name of the tab. A pop-up menu will appear and we can choose to Rename the worksheet.



8.13 Printing

When we want to print a spreadsheet, we must first select the cells that we wish to print. We need to go to the Page Layout tab and select Print Area → Set Print Area. Once the print area is set, click on the icon to bring up the Page Setup dialog box.



This dialog box allows us to scale the printing so that the printed area of the spreadsheet will fit onto a single page, and it allows us to specify the margins, header and footer, and whether the grid lines will be printed or not. We can centre the spreadsheet on the paper and preview the output before we print it.

CHAPTER 9

Databases

9.1 Introduction

Long before computers existed, we have been creating “databases” and using them for keeping important information. People maintain address books, dentists and doctors maintain patient records, libraries have kept catalogues of their books and automobile dealers keep track of who their customers are and when their cars are serviced.

A database is a collection of data about a particular subject. Information is stored so it can be easily retrieved. With the advent of computers, storing and retrieving large amounts of information has become easier. If you want to find a customer by surname, it may not take too long to search through filing cabinets and pull out the correct record. If instead you want to find all of the people who have purchased red cars since 1997 this would be a difficult manual task. But it is easy to do with an electronic database.

There are many ways to store information in a database. The relational database model was introduced in the 1970’s by E. F. Codd. This model allows the user to see a conceptual (logical) view of the data and to ignore the way the data is physically stored on the disk (known as the internal view). The relational model is the most common database model used today.

9.1.1 Databases and Database Management Systems

A *database* is a collection of data, organised in a systematic way. The term database, strictly speaking, refers only to the actual data itself.

A *database management system* is computer software that is used to access and manipulate the data contained in a database.

Although the term “database” should be used only to refer to the data itself and the term “database management system” should be used to refer to the computer program that manages the data, many people refer to the combination of both the data and the software as a database.

9.2 Elements of a database

A relational database consists of a number of tables that are related to each other. Each table contains data organised in rows and columns just like the data in a spreadsheet.

9.2.1 Table

A table is a collection of data organized into vertical columns and horizontal rows. For example, an entire telephone book could be stored as a (very large) table.

Clients

ID	Surname	Given Names	Address	Phone
5	Smith	John	12 Appleton Rd	834-9841
6	Gregorson	Sue	323 Gt North Rd	374-7211
7	Tso	Wen	88 Prince St	883-1220
8	Winters	James	57 Symonds St	376-7749
9	Humfrey	Penny	76 Agria Ave	772-9907
10	Bigwood	Ralph	34 Lesser Dr	874-8858
11	Brothers	Robert	20 Rimu Tce	622-3375
12	Simons	Wendy	95 Segei St	424-4557
13	Lutty	Patrice	81 Tolga Rd	363-7712

9.2.2 Record

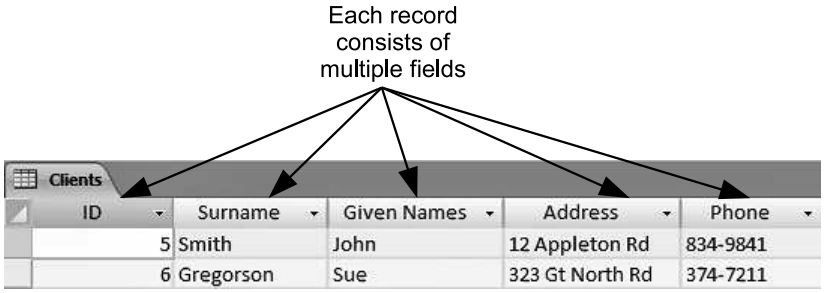
A record is one single row in the table. It typically contains the information about one person, thing or place.

Each row is a record

8	Winters	James	57 Symonds St	376-7749
9	Humfrey	Penny	76 Agria Ave	772-9907
10	Bigwood	Ralph	34 Lesser Dr	874-8858

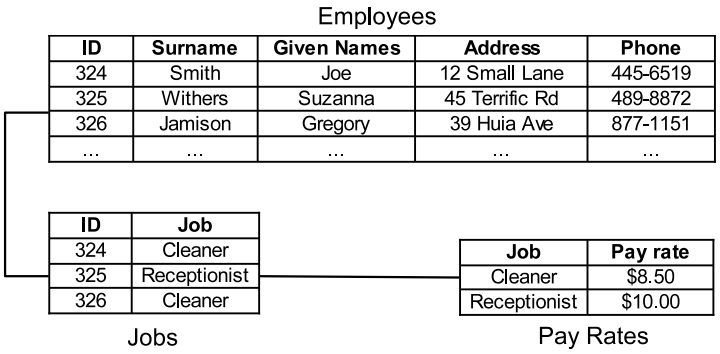
9.2.3 Fields

A field contains a single piece of information. It is the smallest part of information that makes up a record. A single record consists of multiple fields.



9.2.4 Relationships between tables

Both spreadsheets and databases use tables to store data. The main difference is that databases consist of multiple tables that are linked together. For example, the following diagram shows the design for a simple database.



This database consists of three different tables. The first table, called "Employees" contains the personal details of the employees (i.e. their names, addresses, and phone numbers). The second table contains an employee ID and the job title for that employee. The first and second tables are linked together by the ID fields (i.e. an employee with a given ID in the first table refers to the same employee as the ID in the second table). The second and third tables are linked together by the job fields.

9.2.5 What can we do with a database?

Databases are collections of data about a particular subject. We can think of this as an electronic filing cabinet. The user of the database is given a variety of facilities to perform on the information contained in the database. They may:

- Find information in the database
- Add, delete and modify records in the database
- Sort and filter records
- Analyse data

- Generate reports

9.2.6 What are some advantages of databases?

Compactness We no longer need to have rooms full of filing cabinets. The electronic version of this information can be stored on computer disks in a fraction of the space.

Speed A computer can retrieve and update information faster than a human

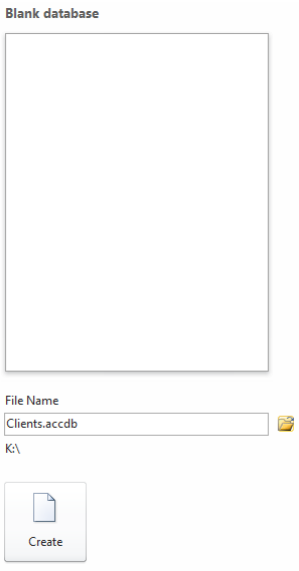
Accuracy We can apply integrity checks to our electronic database to help insure we are entering/storing accurate information.

Standards We can enforce standards in representing information

Security We can apply security restrictions to all or parts of the information contained in the database.

9.3 Creating your own database

To create a new database, choose File tab → New. Before you can start working on the database, you must decide where to save the database file. Use the area on the right side of the window to specify a name and location for your database. When you have chosen the name and location, click on the Create button.

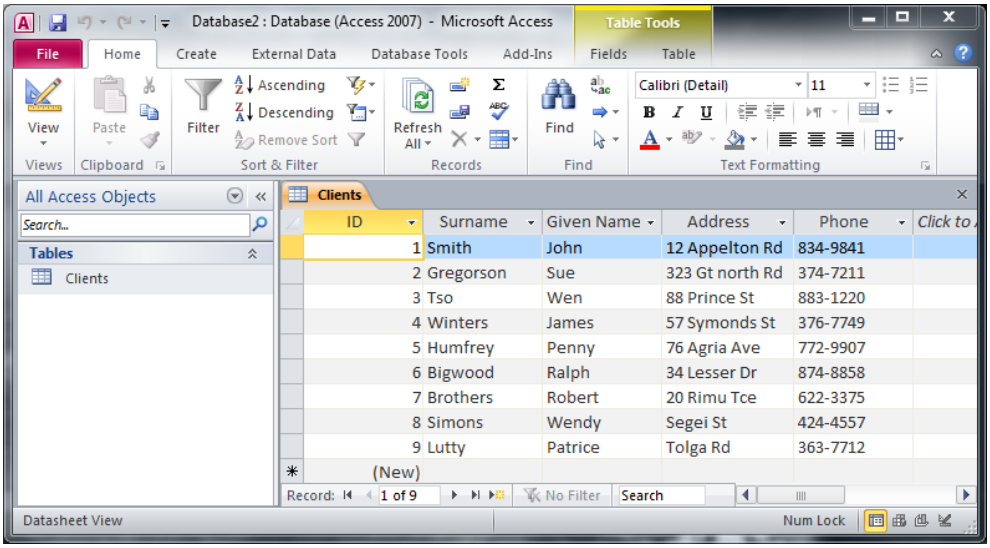


9.3.1 Working with database objects

Access databases consist of many different kinds of objects. The main types of objects are listed below:

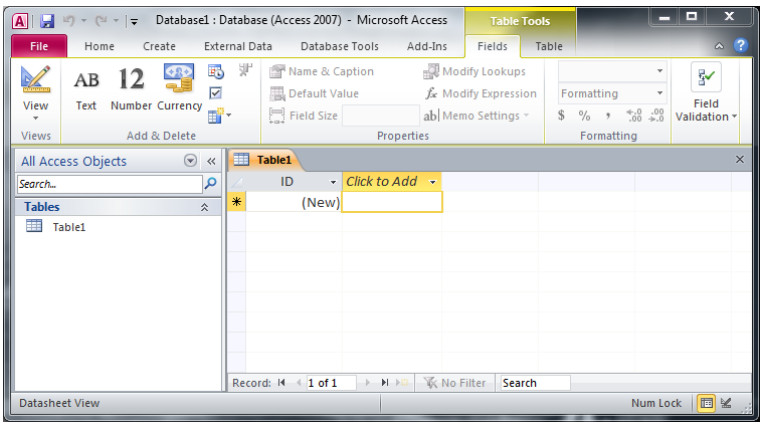
Object	Purpose
Table	Store data
Query	Find and display data
Form	Enter, view and update data
Report	Print summaries of data

Each object that you create will appear in the navigation area to the left. If you select one of the objects, you will see a view of that object in the workspace to the right. In the diagram below, the workspace area is filled with a datasheet showing the contents of a table.



9.4 Tables

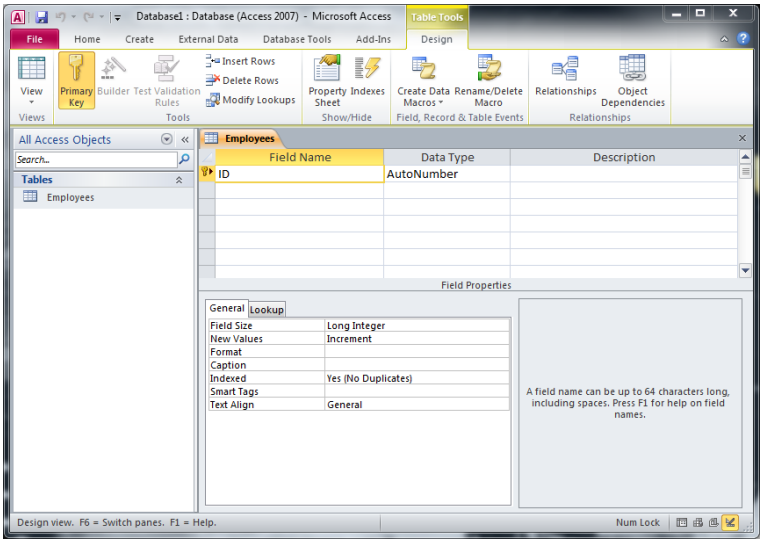
Tables are used to store all of our data. Each column in the table represents a different field. In other words, the columns define the kind of information that we want to store.



By default, the view used to create a table is the datasheet view. Entering data using this view is very similar to using an Excel spreadsheet. You can enter data the way you would in a spreadsheet. To change the name of each field, right-click in the heading for each column and choose to rename.

The datasheet view is good for entering data, but not the best view to use for designing tables. To create the design of a table, click the View button in the Home tab and choose Design View. You will be asked to name the table so that it can be saved before you continue.

9.4.1 Design view



9.4.2 Defining fields

At the top of the workspace is the area where you enter the field name, data type and description. If you right-click in this space you can Insert Rows or Delete Rows. You

can add new fields by clicking in one of the cells in the table and entering the relevant information.

Field names

Each field should have a name. Some names have special meaning to Access and should be avoided (such as “name”, “date”, “off” and “on”). You will receive a warning if you choose a name that is already reserved for use by Access. These field names will be used when the tables and reports are printed, so use the capitalization and spelling that you want to see in a final printed report.

Data types

Each field stores data of a specified type. Some types are very general and others are very specific. The following table will help you choose the most appropriate type of data for each field.

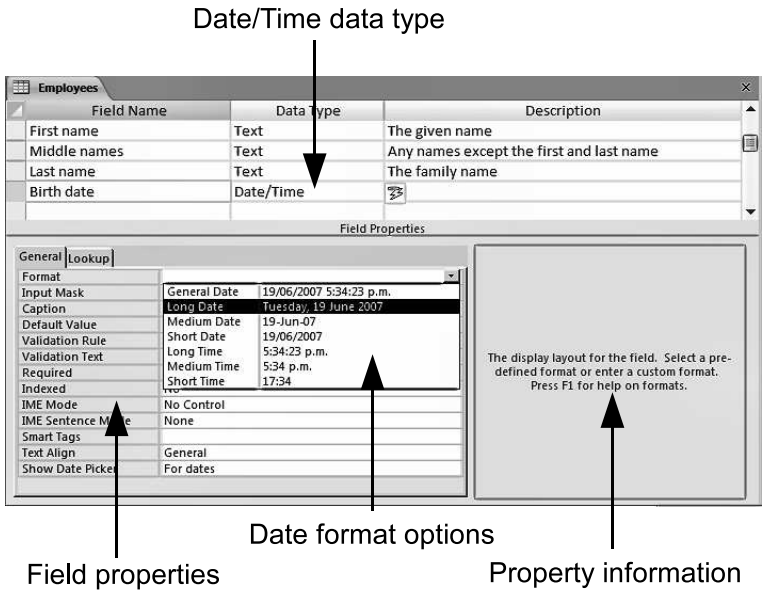
Data type	Purpose
Text	Small amounts of text (255 characters or less)
Memo	Large amounts of text (more than 255 characters)
Number	Numbers that are used in calculations
Date/Time	Dates and times
Currency	Numbers that represent currency
AutoNumber	Unique sequential numbers (e.g. ID)
Yes/No	Data that has only two alternatives (e.g. T/F)
OLE object	Image, sound, graph or document
Hyperlink	Link to a destination
Attachment	One or more files

9.4.3 Field descriptions

Each field has a description associated with it. This description will not be printed, and will not form part of any final report. However, it is useful to remind users of the database about the kind of information that should be put into each field.

9.4.4 Field properties

The lower area of the workspace lists the properties for each field. If you click on each property in turn the purpose of the property will be displayed in the information area to the right of the property.



Format

The format property can be used to change the way that information in the field is displayed.

Symbol	Purpose
@	Expects a text character (or space) in this position.
&	Text character in this position is optional.
>	Forces all characters to uppercase.
<	Forces all characters to lowercase.

The following examples show how you could use the format property to display data in a specific way. More information is available from Access online help.

Symbols	Original data	Displayed
(@@) @@@-@@@@	094458783	(09) 445-8783
@@@@@@@@@@@	09-445-8783	09-445-8783
	094458783	094458783
>	Access	ACCESS
	access	ACCESS
<	Access	access
	ACCESS	access

InputMask

The input mask property allows you to control what the user can enter into a field, and to provide guidance to the user to make the data easier to enter. Essentially, an input

mask defines a "rule" that specifies what data is accepted in this field.

There are many different options for the input mask. See the online help for "InputMask Property" for more information. Some of the options are shown in the following table.

Character	Meaning
0	Digit [0–9] required.
9	Digit [0–9] or space allowed, but not required.
L	Letter [A–Z] required.
?	Letter [A–Z] allowed, but not required.
A	Letter or digit required.
a	Letter or digit allowed, but not required.
&	Any character or space required.
C	Any character or space allowed, but not required.

The following examples show how you could use the input mask property to only allow specific data to be entered. More information is available from Access online help.

Input mask	Sample data
(99) 000-0000	(09) 445-6298 () 445-6293
(99) 000-0000 ? 99999	() 387-2246 (09) 373-7599 x 85654 () 811-3352 x 326
>L<??????????	Andrew

Note: If you are limiting the input in this way, make sure that *all* the data that you will want to input will be accepted. This is especially important with first names and surnames.

Default value

If a default value is set, then it is automatically entered into the field when a new record is created. The user can accept the default value or change it to a different value. If the value is unlikely to change from the default value, then this saves time when you create new records during data entry.

Validation rule

If you want to restrict the input values more than what is possible using the input mask, then you can use a validation rule. This is an expression which checks to see if the input is valid or not. If the input is not valid, then a message can be displayed to the user to let them know what is wrong.

You can use any of the following comparison operators in the validation rule:

Symbol	Meaning
=	Equal to
<>	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

You can also use the following logical operators to join two rules together:

Symbol	Meaning
OR	Accept input if it fulfils either the first or the second rule
AND	Accept input only if it fulfils both the first and second rule

For example:

Expressions
"F" OR "M"
>= 12
>= 18 AND <65

9.4.5 Primary keys

In each table, one of the fields should be defined as a "primary key". This field is used to uniquely identify a record in the table. It is also used to link data from different tables together. The data in a primary key must be unique for each record. For example, an IRD number, your driver licence number, a car number plate, a student ID, or a part number in a catalogue are all examples of data that would be used as a primary key.

A primary key prevents duplicate entries from being entered in the table. A primary key should never be changed once entered, and must not be empty (or NULL).

The primary key of a table will be indicated by a little picture of a key appearing next to the field that is used as the primary key.

To set a field as the primary key, right click on the field and choose Primary Key.

9.4.6 Foreign key

A foreign key is a field in a table that relates to the primary key of another table. This field must contain a value found in the primary key of the table to which it relates, or it must be null.

9.4.7 Entering data

Once the table has been created, you can enter data into the table. The datasheet view is most appropriate for data entry. To change to the datasheet view, select the Home tab, click on the View button and select Datasheet View.

9.4.8 Creating more than one table

To create additional tables, go to the Create tab and click on the Table button. The new table will appear in a separate tab within the workspace, and the name of the table will be listed in the navigation area on the left side of the window.

9.4.9 Lookup fields

A lookup field allows you to connect the information in one table with the information in another table.

For example, you might have all the personal information about employees in a table that uses "Employee ID" as the primary key. In a separate table, you might store a list of car-parking spaces and the employees that are allocated different spaces. The tables might look like the following:

Employees

ID	First name	Middle name	Last name	Birth date
34244	James	Eugene	O'Neil	18/03/70
35229	Sarah	Ann	Harris	21/09/81
77212	Josey	Prince	Whitehouse	02/05/64

Carparking

Carpark code	Employee ID
A11	34244
A12	34244
A13	34244
A15	35229

In this case, the employee with ID 34244 (i.e. James O'Neil) has been allocated 3 different carparking spaces.

Note that the *Employee ID* in the *Carparking* table should only contain values chosen from within the *Employees* table. In other words, we don't want to allocate a carparking space to an employee that doesn't exist.

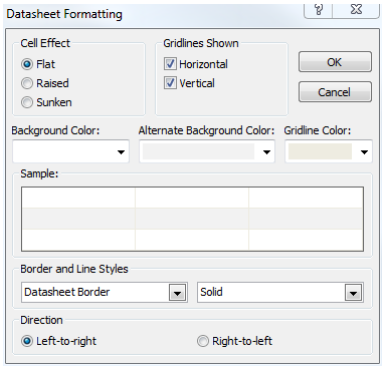
To create a connection between these two tables that ensures we can only choose legitimate values for the *Employee ID* field in the *Carparking* table, we need to set the data type of that field to a Lookup Value. Select the datatype field of the Employee ID field and choose Lookup Wizard. The wizard helps to select the data you want to use for this field. You need to make the following choices:

- Look up the values in a table. Click Next.
- Select the *Employees* table. Click Next.
- Select the *ID* field from the available fields area and click the > button to move the field into the selected fields area. Click Next.
- Choose the order used to present the entries. Select the first field and choose ID. Click Next.
- Adjust the width of columns that will be shown. Click Next

Once you have created a connection between these two tables, you will not be able to enter any values into the *Employee ID* field in the *Carparking* table, rather, you will be able to select the employee ID you want from a drop-down list obtained from the *Employees* table.

9.4.10 Formatting datasheets

From the Home tab, click on the bottom right corner of the *Text Formatting* section to display the Datasheet Formatting window. Using the options in this window, you can change the appearance of the datasheet.



9.5 Forms

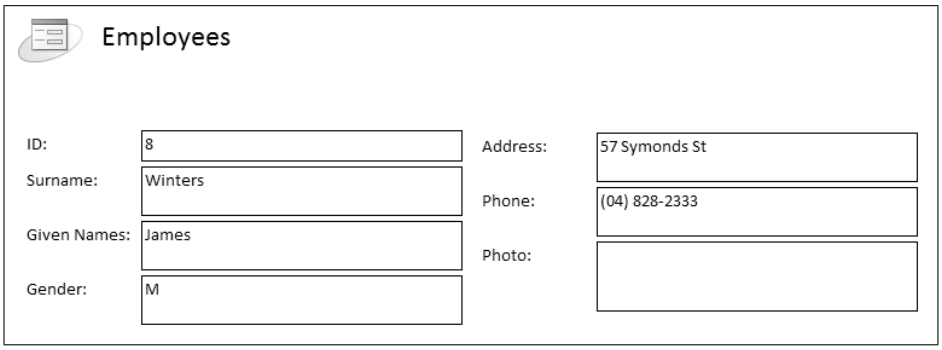
Forms are a way to create a special interface that can be used to create, view and modify records of the database. Since you can decide which fields to display in a form, it can be used to limit access to information. This can make it easier to use the database since users aren't overwhelmed with a lot of information that they don't need to see.

There are a number of different methods that you can use to create a form.

9.5.1 Form tool

The form tool allows you to quickly create a form, but it is not the most versatile approach. Go to the Create tab and click on the Form button. All of the fields will be

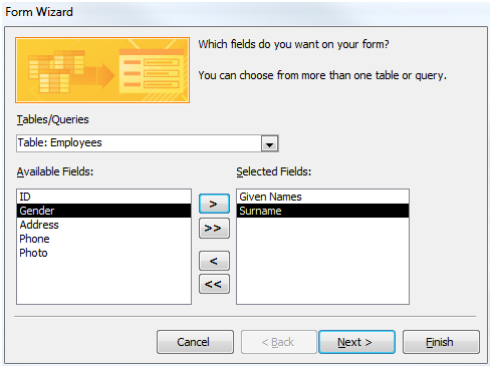
added to the form. You can remove fields that you don't want displayed by selecting the field and choosing delete. You can also rearrange the order of the fields by clicking and dragging them to a new position.



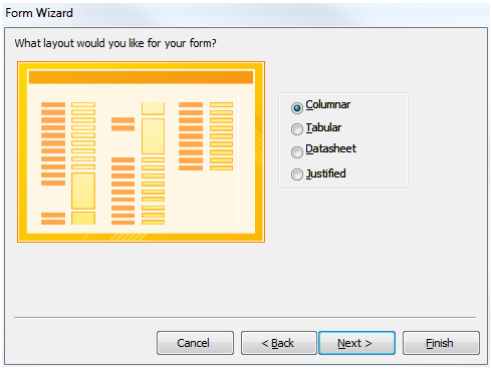
9.5.2 Form wizard

The form wizard gives you a lot of control over the appearance of the form. It is a good method to use when you are starting out with forms. To create a new form using the wizard, go to the Create tab and select Form Wizard.

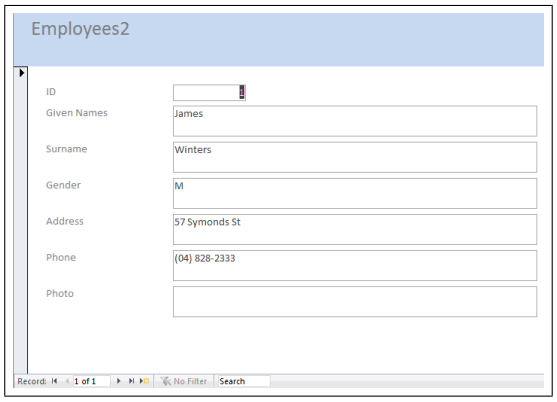
Add the fields that you want to use on the form in the order that you want them to appear.



Choose the layout that you want to use for the form.



Finally, you will have to choose a name for your form, then it will be displayed as an object in Access.



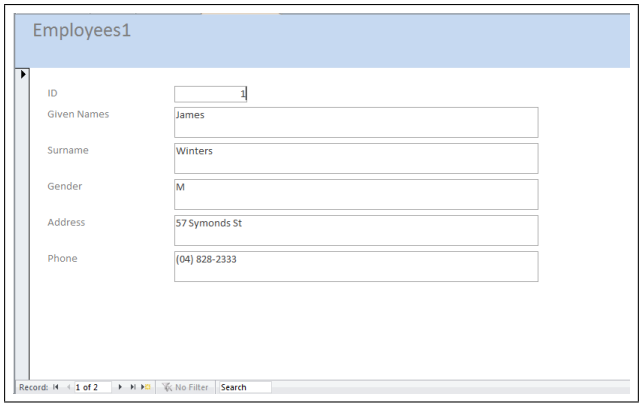
9.5.3 Navigating through forms

You can use the up and down arrow keys to move from one field to the next (or use the Tab key). The Page Up and Page Down keys move to the previous and next record respectively. You can also click on the navigation tools that appear at the bottom of the window to move between different records.

9.5.4 Changing the layout

To change the layout of a form, go to the Home tab, click the View button and choose Layout View. In this view, you can edit the position and appearance of each field on the form.

The fields will be connected together when you start editing the layout. You can shift the ordering of the fields, and you will be able to move the field to a completely new cell in the layout grid. To extend the layout you can add cells using the Insert Above, Insert Below, Insert Left and Insert Right buttons in the Rows & Columns section on the Form Layout Tools tab.



9.6 Queries

A query is a way of requesting data from the database. Once the data has been requested, the results are presented in a table which can be used just like any other table in a database (i.e. you can format and even search the results of the query).

The easiest way to query the database is using the query wizard. Go to the Create tab and click on the Query Wizard. Choose to use the *Simple Query Wizard*.

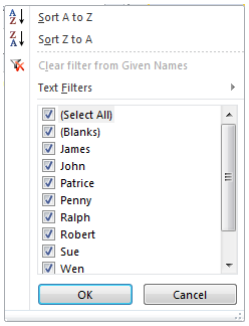
You will be asked to select the fields that you want to use in the query. Select all the fields that you want to display, and all the fields that are involved in your query. For example, if you want to display the names and phone numbers of all the female employees, then the fields you need to add to the query are: Given Names, Surname, Phone and Gender.

Given Names	Surname	Phone	Gender
John	Smith	(02) 838-2323	M
Sue	Gregorson	(09) 336-2519	F
Wen	Tso	(02) 553-6773	M
James	Winters	(04) 828-2333	M
Penny	Humfrey	(04) 662-9798	F
Ralph	Bigwood	(09) 642-2001	M
Robert	Brothers	(09) 428-7572	M
Wendy	Simons	(09) 373-7599	F

Once you have a table containing the fields you are interested in, you can choose to filter the results.

9.6.1 Filtering results

To filter the records that are displayed, click on the name of a field. A dialog box will appear containing all the different values that have been entered for that field in the database. Only the records containing the values that you have checked will be displayed, the others will be hidden from view.



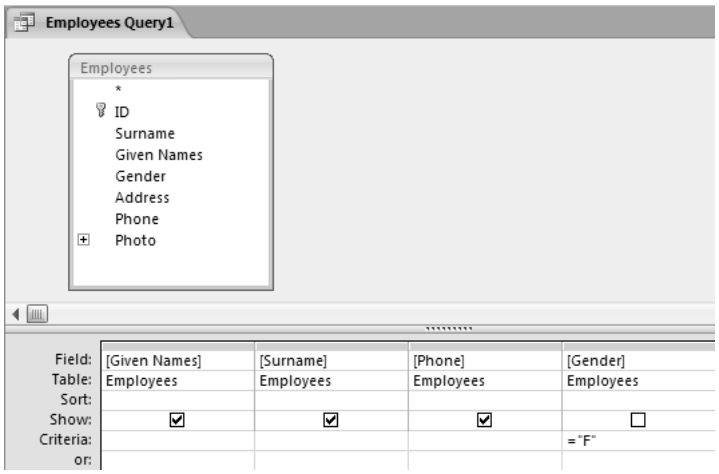
If we only wanted to display the records for females,we would make sure only the “F” is checked in the filter for the *Gender* field. The results of applying this filter are shown below.

Given Names	Surname	Phone	Gender
Sue	Gregorson	(09) 336-2519	F
Penny	Humfrey	(04) 662-9798	F
Wendy	Simons	(09) 373-7599	F

9.6.2 Query Design view

For more precise control over the records that are returned as a result of a query, we can use the query design view to refine the query that we use. If you want to perform more complex queries, you will have to use the Query Design View.

The design view allows you to specify which of the fields will be displayed, and which ones are used in the query, but are hidden. You can also specify the criteria that are used to select the records that you are interested in. In the following example, only the records that have “F” in the *Gender* field will be displayed.



Although the Gender field is used in the query, the box indicating whether or not to show the field has not been checked so the Gender field will not be displayed in the results of the query. The results from this query are shown below.

Given Names	Surname	Phone
Sue	Gregorson	(09) 336-2519
Penny	Humfrey	(04) 662-9798
Wendy	Simons	(09) 373-7599

The criteria used in this example is

- =“F”.

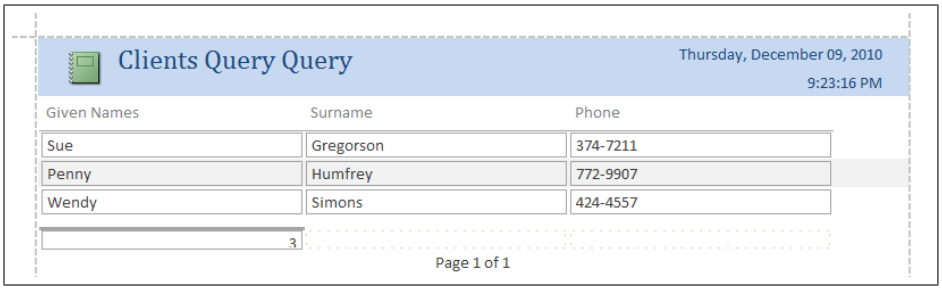
The format for specifying criteria is the same as for validation rules. See section 9.4.4 for more information.

9.7 Reports

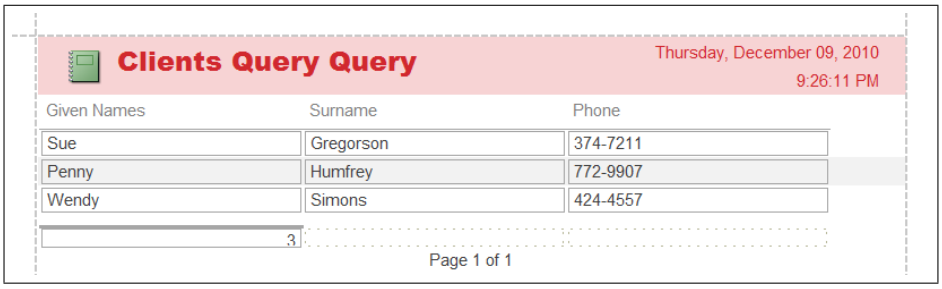
A report is used to create output from a database that looks attractive and professional. There are a variety of different ways that reports can be created.

9.7.1 Report tool

The report tool is the easiest way to generate reports. First, open the table that you want to create a report from. This can be a table of data, or (more often) it is a table containing the results of a query. To create the report, go to the Create tab and click the Report button. The contents of the table will be displayed in a report.



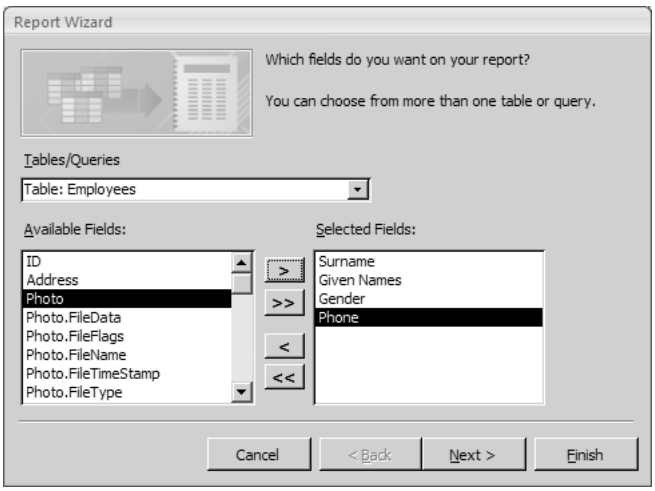
The appearance of the report can be changed by clicking the Themes button on the Report Layout Tools → Design tab.



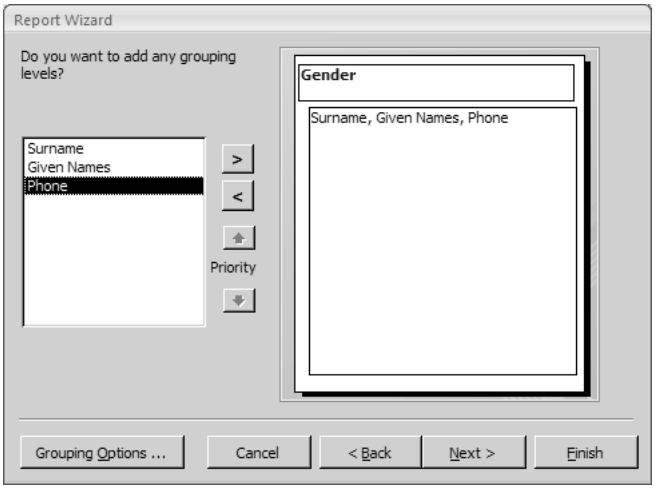
9.7.2 Report wizard

The report wizard allows you more control over the fields that are used in the report and how the report is arranged on the page. To create a report using the report wizard, go to the Create tab and click the Report Wizard button. Follow the dialog boxes to generate a report.

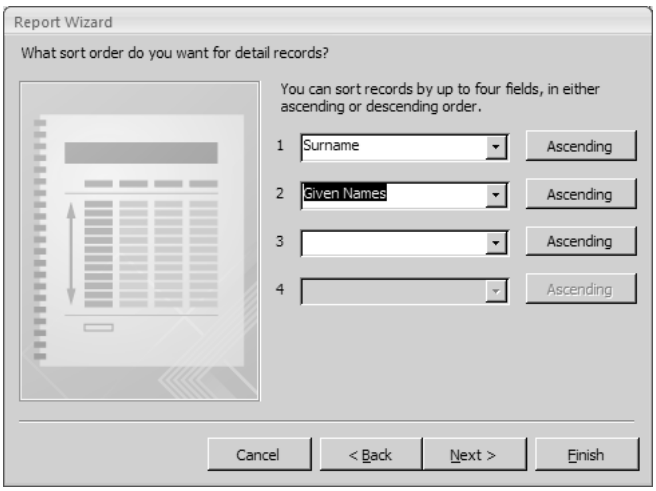
Select the fields that you want to appear on the report.



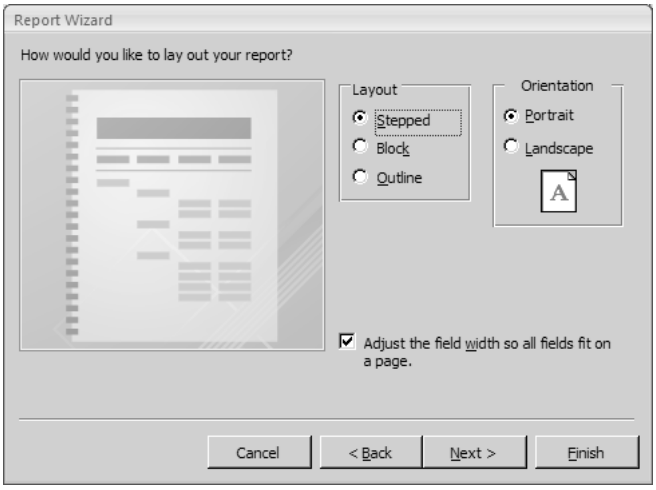
Add grouping levels if you want. If you add a grouping level, then the records will be grouped according that field.



The order used to display the records is decided by the sort criteria. Select the fields used to sort the records.



Choose the overall layout. Columnar is the most common layout used.



The final report is ready for printing. The print preview view shows what the report will look like when it is printed.

Employees2

Gender	Surname	Given Names	Phone
F			
	Gregorson	Sue	(09) 336-2519
	Humfrey	Penny	(04) 662-9798
	Simons	Wendy	(09) 373-7599
M			
	Bigwood	Ralph	(09) 642-2001
	Brothers	Robert	(09) 428-7572
	Lutty	Patrice	(03) 335-2390
	Smith	John	(02) 838-2323
	Tso	Wen	(02) 553-6773
	Winters	James	(04) 828-2333

9.7.3 Design View

For full control over the design of a report, you will need to use the Design View. If you have created a report using the report wizard, then the design view will allow you to make minor adjustments before printing.

9.8 Structured Query Language (SQL)

In order to query our databases we must use a computer language. The most widely used computer language for querying databases is called Structured Query Language or SQL. This language is small and simple, yet allows us to perform all of the necessary operations to add, delete, update and modify tables in a database.

9.8.1 SELECT

The SELECT command is used to retrieve lists of records from a specified database table. We can specify the fields that we wish to be returned. When we want to retrieve multiple columns, we must specify each field name. A comma must be used to separate field names. The columns will display in the order you select them.

The format for this is:

```
SELECT fieldname, fieldname, fieldname
FROM tablename;
```

9.8.2 ORDER BY

The ORDER BY clause tells SQL you want the specified fields displayed in ascending order (ordered from A–Z, 1–100). Notice that this command is exactly the same as before, but with the ORDER BY clause added.

```
SELECT fieldname, fieldname
FROM tablename
ORDER BY fieldname;
```

9.8.3 WHERE

The WHERE clause is used to constrain the records from which fields will be selected. The only records for which fields will be returned are the ones that fulfil the requirements listed after the WHERE clause.

If you are referring to text values in the WHERE clause, then they must be surrounded by single quotes. If you are referring to numeric values, then they should not be surrounded by quotes.

```
SELECT fieldname, fieldname,.. fieldname
FROM tablename
WHERE fieldname logical operator value
```

CHAPTER 10

Python

10.1 Computer programming

A computer program is a sequence of instructions that tells the computer how to perform a specific task. These instructions are written using a formal computer language. There are many hundreds of different computer languages. The language that we will be using is called Python. It is a high-level language, similar to most modern languages that computer programmers use to develop the computer software we are familiar with.

Python is an interpreted language. In this kind of language, a computer program called an interpreter is used to convert the instructions that we use (Python instructions) into instructions that the CPU uses (machine code).

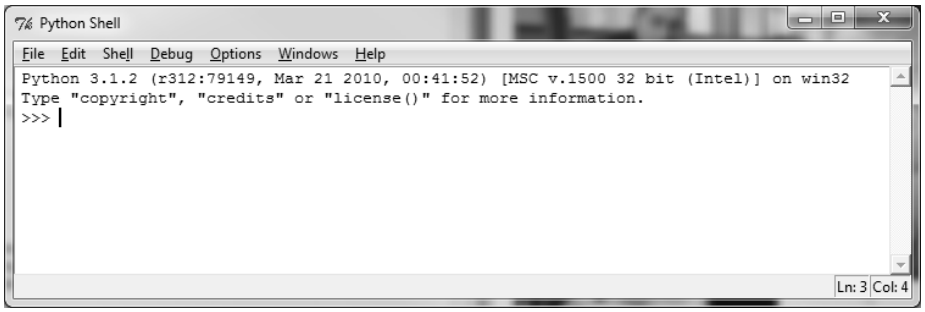
10.2 Using IDLE to program in Python

A program called IDLE has been developed to make it easier for people to use the Python programming language on the Windows operating system. IDLE is a simple Integrated Development Environment. That means it contains an editor that a programmer can use to write their programs. It also contains tools that allow the programmer to test, debug and run their programs.

Since Python is an interpreted language, we have two different options when we want to run a Python program. We can run the instructions as we type them, or we can type an entire program out and run it all at once. Each option is described in more detail below.

10.2.1 Using an interactive interpreter

The IDLE program includes an interactive interpreter. When we start IDLE, we see a window that contains a prompt as shown in the screenshot below:



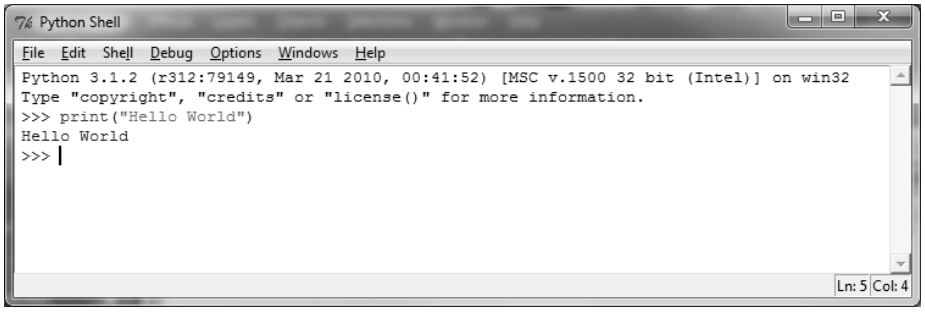
We are able to type Python instructions at the prompt and when we hit the Enter key, it will immediately execute the instruction and show us the results. In other words, the language is being interpreted interactively (the commands are being interpreted as we type them).

By tradition, the first program that we usually write when we learn a new computer language is the “Hello World” program. This program simply prints out the text “Hello World” to the screen. To write this program in Python, we use the instruction:

Python Source Code

```
1 print ("Hello World")
```

The interpreter will immediately follow the instruction and print the result as shown in the screenshot below:



This is an excellent way to try out new instructions and to test single lines of code to see what they do. However, for a program that consists of many instructions, it is best to write the program in a separate file, save the file to disk and run the entire program at once.

10.2.2 Writing a Python program

A Python program can be written in any text editor. Once the file is saved to disk, it is possible to run the Python program by double-clicking on the file on the desktop, just

like a normal windows application. However, we will use the editor that comes with the IDLE application to both write our Python programs and also to run them.

10.3 Statements

A single instruction in a programming language is called a *statement*. A program consists of a sequence of statements that are executed in order, one after another. This chapter will concentrate on the most common kinds of statements that are used in Python.

10.4 Comments

A comment is used to include messages in the source code of a program that other humans can read, but which the computer will ignore. For example, most programmers write their name and the date at the start of the program so that other people know who wrote the program and when it was written. These comments are completely ignored by the computer.

To include a comment in a Python program, you should start the line with a hash sign (#). Any text that appears after the hash sign will be ignored until the end of the line is reached. An example of comments in a Python program is:

Python Source Code

```
1 #Author: Andrew Luxton-Reilly
2 #Date: 01/01/06
3 #Purpose:  This is a short program used to calculate
4 #          the average of a sequence of numbers
```

Every program that we write should include the name of the author as a comment at the top of the file. The comment that should appear at beginning of our programs will not be included in most of the examples listed in this chapter in order to save space. However, when you write your own programs you should always include a comment that contains your name at the top of the file.

10.5 A first program

It is traditional for programmers to write a “Hello World” program whenever they start using a new computer language. We will continue this tradition when we learn Python. The first Python program we should write is as follows:

Python Source Code

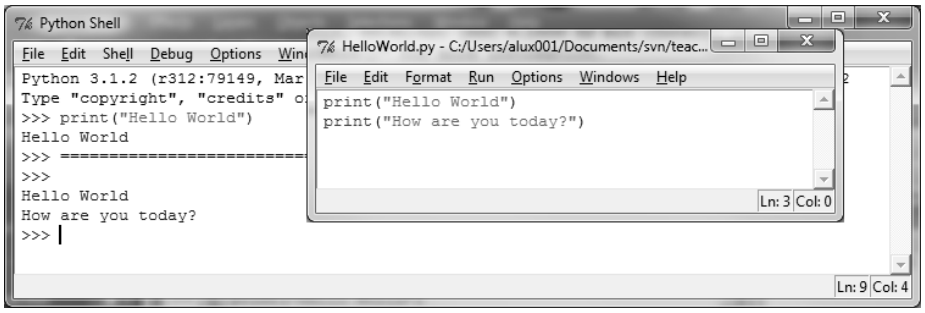
```
1 print ("Hello World")
```

Once we have entered this text, we should save the program on disk as HelloWorld.py. The names of any file that contains a Python program should end with the file extension .py.

Once the program is saved, we can run the program by choosing Run → Run Module. The output that appears is:

```
Hello World
```

The following screenshot shows how this looks in the IDLE application. The source code (containing the Python program) appears in its own window. The output of the program will be shown in the main Python Shell window.



10.6 Printing

One of the most common statements in the Python language is the `print` statement. This statement is used to print information to the screen. We can print numbers or text using this statement. For example, the following Python program:

Python Source Code

```
1 print ("The current year is")
2 print (2006)
```

will produce the output:

```
The current year is
2006
```

If we want to print more than one thing on the same line, then we can separate the things that we want printed with a comma. For example, the program:

Python Source Code

```
1 print ("The current year is", 2006)
```

will produce the output:

```
The current year is 2006
```

10.7 Strings

When we print out text, we are printing a sequence of characters. In computer programming, we normally refer to a "sequence of characters" as a "string of characters". For this reason, programming languages today refer to a sequence of characters as a "string".

In Python, a string must be enclosed in quote marks. We can use either single quotes or double quotes to enclose a string. For example, the program:

Python Source Code

```
1 print ("This is a string enclosed in double quotes.")
2 print ('This is a string enclosed in single quotes.')
```

will produce the output:

```
This is a string enclosed in double quotes.
This is a string enclosed in single quotes.
```

It is worth noting that a string is treated differently to a number. The string `"234"` is treated as the character 2, followed by the character 3, followed by the character 4. These characters are normally represented by ASCII or Unicode values, and are treated by the computer the same as any other character typed in from the keyboard. In other words, they are treated as plain text.

In this chapter, we will use double quotes to enclose all the strings we use.

10.8 Numbers

In most programming languages, numbers come in two different varieties. A number can either be an integer, or it can be a floating-point number. The distinction is made between integer and floating-point numbers because computer hardware is designed to treat these kinds of numbers differently (i.e. one part of the CPU is used to perform calculations between integer numbers and a different part of the CPU is used to perform calculations between floating-point numbers).

An integer number is any number that *does not* have a decimal point. It can be positive, negative or zero. For example, the following numbers are all integer numbers:

```
-1890, 0, 34, 89200193
```

A floating-point number is any number that *does* have a decimal point. It can be positive, negative or zero. For example, the following numbers are all floating-point numbers:

```
-33.3333333333, -12.75, 0.0, 2.5, 7.0, 34.779, 0.0000000001
```

Note that a floating-point number is permitted to have a zero after the decimal point.

10.8.1 Printing numbers

We can print these numbers out using a print statement. For example, the following program:

Python Source Code

```
print (5)
print (7.5)
print (0.000001)
```

will produce the output:

```
5
7.5
0.000001
```

Note that numbers never have quote marks surrounding them. If they had quote marks, then they would be treated as strings, not as numbers.

10.9 Mathematical operations

Python supports all the normal mathematical operations. The following table summarizes the kinds of operations that apply to numbers, and the operators that are used in Python to represent these operations.

Operation	Symbol	Example	Result
Exponent	**	2 ** 3	8
Multiply	*	3 * 4	12
Divide	/	10 / 2	5.0
Add	+	3 + 5	8
Subtract	-	4 - 7	-3
Remainder	%	15 % 6	3

These operations can be applied to either integers or floating-point numbers. However, if both of the operands (the things that the operator applies to) are integers then the result will also be an integer. If at least one of the operands is a floating-point number, then the result will be a floating-point number. For example:

```
2 + 3    = 5
2 + 3.0  = 5.0
2.0 + 3  = 5.0
2.0 + 3.0 = 5.0
```

Note that in the case of division, the result is always a floating point number. For example:

```
9 / 10  = 0.9
10 / 5 = 2.0
10 / 8  = 1.25
```

If we always want to keep only the integer part of a division (throwing away the fractional part), then we can use the special integer division operator `//`. For example:

```
3 // 4  = 0
7 // 3 = 2
```

The remainder operation (`%`) will give the remainder after the first operand is divided by the second. For example:

```
8 % 4 = 0
9 % 4 = 1
10 % 4 = 2
11 % 4 = 3
12 % 4 = 0
```

To visualise what is happening with the remainder operation, it might help to imagine that we are distributing a number of things among a group of people. If we had 11 cars and 4 people, then we can give each person 2 cars and we would be left with 3 cars. The `%` operator gives us the number of cars left over, in other words, the remainder.

If the operation is inside quote marks, then it will be treated as a simple character that forms part of a string. However, if Python encounters any of these mathematical operations outside a string, it will perform the calculation and use the result. For example, running the program:

Python Source Code

```
1 print ("2 + 3 =", 2 + 3)
2 print ("2 * 3 =", 2 * 3)
3 print ("3 - 2 =", 3 - 2)
4 print ("3 / 2 =", 3 / 2)
5 print ("3 ** 2 =", 3 ** 2)
6 print ("3 % 2 =", 3 % 2)
```

will result in the output:

```
2 + 3 = 5
2 * 3 = 6
3 - 2 = 1
3 / 2 = 1.5
3 ** 2 = 9
3 % 2 = 1
```

10.9.1 Order of precedence

When we are faced with an expression that has a number of different operators, we have to decide which operator we will apply first. We are taught in mathematics that we should always apply brackets first, then exponents, then multiplication and division from left to right, then addition and subtraction from left to right. The same order applies to expressions in programming languages. For example, when faced with an expression such as:

```
(2 + 3) * 2 ** 2 / 2 - 1 + 2
```

we normally evaluate the brackets first:

```
5 * 2 ** 2 / 2 - 1 + 2
```

and then any exponents

```
5 * 4 / 2 - 1 + 2
```

and then multiplication and division from left to right. In this case the multiplication is furthest to the left

```
20 / 2 - 1 + 2
```

and then the division is evaluated

```
10.0 - 1 + 2
```

followed by addition and subtraction from left to right. In this case the subtraction is furthest to the left

```
9.0 + 2
```

and finally the addition

```
11.0
```

The Python programming language will follow the same rules. It is worth noting that the remainder operation `%` has the same priority as multiplication and division. If you are not sure when you are writing code, then it is always wise to use additional brackets to make the order explicit.

10.10 String operations

Although we are used to applying mathematical operators to numbers, we do not usually apply them to text. However, most programming languages allow you to apply operators to strings as well as numbers, and the symbols used in mathematics are often used for this purpose.

If we want to join two strings together, we can use the `+` operator. If we want to repeat a string a number of times, we can use the `*` operator. The following table summarizes these operations:

Operation	Symbol	Example	Result
Concatenation	`+`	`"hello" + "world"`	`"helloworld"`
Repetition	`*`	`"hello" * 3`	`"hellohellohello"`

For example, the following program:

Python Source Code

```
1 print ("echo" + " and " + "echo")
2 print ("echo" * 4)
```

will result in the output:

```
echo and echo
echoechoechoecho
```

Note that the repetition operator is applied before the concatenation (we do * before +). The following program:

Python Source Code

```
print ("Mary had a " + "little lamb, " * 3)
print ("Mary had a " + "little lamb, ")
print ("Her fleece " + "was white as snow.")
```

will produce the output:

```
Mary had a little lamb, little lamb, little lamb,
Mary had a little lamb,
Her fleece was white as snow.
```

10.11 Variables

A variable in mathematics is a symbol that stands for a number. It acts as a placeholder for that number and it means the same thing wherever it is encountered. It would be impossible for the following to be true in mathematics:

```
x = 7
x = 3
```

However, in programming, the word *variable* has a completely different meaning. In programming, a variable is a box that is used to store information. A given box can store different information at different times, so the following could be true in Python:

```
x = 7 #store the value 7 in the box called x
x = 3 #store the value 3 in the box called x
```

Each variable needs a name so that we can tell them apart. In Python, there are rules about what kind of names we are allowed to use.

- A variable name must start with a letter
- A variable name may only contain lower case letters, digits, or the underscore character.

The convention in Python is that variable names start with a lower case letter. Variable names can contain digits, just not as the first character of the variable name. If the name is a combination of words, then each word should be separated with an underscore character. Some examples are: `my_age`, `number_of_weasels`, `size_of_rectangle`.

10.11.1 Assigning a value to a variable

A variable is no use to us unless we know how to store information in the variable, and how to get information back out of the variable. A special operator is used to store information into a variable. The operator is called the *assignment* operator, and is represented by the equals sign in Python (=). For example, the Python code:

Python Source Code

```
1 age = 34
```

will take the number 34 and it will store that number in a box called `age`.

Note that the equals sign does not mean the same as it does in mathematics. We are not saying that these two things are equal, rather we are saying "Take the value on the right hand side of the equals sign and *store* that value in the *variable* named on the left hand side of the equals sign".

10.11.2 Using the value stored in a variable

In order to get information out of the box, we simply use the name of the variable and Python will look inside the box with that name and use the contents. For example, the Python program:

Python Source Code

```
1 age = 34
2 print (age)
```

will result in the output:

```
34
```

10.11.3 Assignment happens last

When we use the assignment operator, it is applied after all the other operators that appear in that line of code. In other words, we do any calculation that appears to the right of the = operator before the assignment takes place. For example, the following program:

Python Source Code

```
1 height = 34
2 width = 10
3 area = height * width
```

will result in the value `340` being stored in the variable called `area`. That is because the multiplication (`height * width`) is done before the assignment operation(=).

The following program is used to keep a running total of a series of numbers:

Python Source Code

```
1 sum = 0
2 sum = sum + 27
3 sum = sum + 4
4 sum = sum + 10
5 print (sum)
```

We will look at this program in detail below:

line 1 The value `0` will be stored in a variable called `sum`

line 2 We start by evaluating the expression `sum + 27`. The value stored in `sum` (which is `0`) is added to `27`, so the result is the value `27`. This result is stored into the variable called `sum`.

line 3 We start by evaluating the expression `sum + 4`. The value stored in `sum` (which is now `27`) is added to `4`, so the result is the value `31`. This result is stored into the variable called `sum`.

line 4 We start by evaluating the expression `sum + 10`. The value stored in `sum` (which is now `31`) is added to `10`, so the result is the value `41`. This result is stored into the variable called `sum`.

line 5 The value stored in `sum` is printed out. This value is currently `41`.

The output from this program will be:

```
41
```

10.12 Reading input from the user

To read input from the user, we use the `input(`**prompt**`)` function. Note that this function will print out the prompt to the screen, and then wait for the user to enter some data. When the user presses the Enter key, then the program will read the information that they have entered and continue to execute the statements in the program.

When we use the `input()` function, we need to store the information that the user enters in a variable, otherwise the information would be lost and we would not be able to use it later in the program. The following program shows how you would normally use the function to read from the user:

Python Source Code

```
1 name = input("Enter your name: ")
2 print ("Hello", name)
```

If we want to read an integer from the user, we enclose the `input( )` function in the `int( )` function that converts the input into an integer. The typical use of this would be: `int( input(`**prompt**`))`. For example:

Python Source Code

```
1 age = int(input("Enter your age: "))
2 print ("Next year, you will be", age + 1)
```

If we want to read a floating point number (i.e. a number containing a decimal point) from the user, we enclose the `input( )` function in the `float( )` function that converts the input into a floating point number. The typical use of this would be: `float( input(`**prompt**`))`. For example, if we wanted to write a program that converts a value from miles to kilometres, we could use the formula:

$$kilometres = 1.609344 \times miles$$

The Python program should do the following:

- Ask the user to enter a number of miles
- Convert the number of miles to a number of kilometres
- Print the output to the screen

Since we would want to read in the number of miles with decimal points, we would use the following code:

Python Source Code

```
1 #Author: Andrew Luxton-Reilly
2 #Date: 6/05/06
3 #Purpose: Convert miles to kilometres
4
5 miles = float(input("Please enter the number of miles: "))
6 kilometres = 1.609344 * miles
7 print (miles, "miles is", kilometres, "kilometres")
```

The output from this program would be:

```
Please enter the number of miles: 40.5
40.5 miles is 65.178432 kilometres
```

10.13 Making Decisions: if, elif, and else statements

An `if` statement is used to allow our programs to make decisions. This is very similar to the `IF` functions that are used in spreadsheet formulae. We use an `if` statement to test a specified condition to see if that condition is true or false. If the condition is true then we execute one set of statements; otherwise these statements are not executed.

The format for the `if` statement is more complex than any of the statements we have looked at so far. The general format is as follows:

```
if <condition>:
    statements to execute if condition is true
```

There are some important notes to remember about the format of an `if` statement:

- The `<condition>` must be a boolean expression (i.e. an expression that evaluates to true or false).
- There is a colon (`:`) after the condition.
- The `statements to execute if condition is true` are a block of statements that are executed if the condition evaluates to true. They must be *indented*.

- The blocks of statements must be indented to the same level (i.e. the same number of spaces must precede all statements in the block). Do not use a mixture of Tab and Space to indent. Either use all spaces, or use all tabs.

The following program illustrates how an `if` statement can be used:

Python Source Code

```
#Author: Damir Azhar
#Date: 23/11/15
#Purpose: To decide if a person is able to vote or not

age = int(input("Please enter your age: "))
if age < 18:
    print ("You are too young to vote")
if age >= 18:
    print ("You can legally vote")
```

When the user enters the number `16` at the prompt, the output from this program is:

```
Please enter your age: 16
You are too young to vote
```

When the user enters the number `23` at the prompt, the output from this program is:

```
Please enter your age: 23
You can legally vote
```

10.13.1 If...Else Statement

In the previous example we saw how two `if` statements can be used to decide between two scenarios: the user is younger than 18 (and is too young to vote), or the user is 18 or older (and is eligible to vote). We can also see that these two scenarios are all encompassing; every possible age is taken into account.

Instead of using two `if` statements we can use an `if...else` statement as follows:

Python Source Code

```
#Author: Damir Azhar
#Date: 23/11/15
#Purpose: To decide if a person is able to vote or not

age = int(input("Please enter your age: "))
if age < 18:
    print ("You are too young to vote")
else:
    print ("You can legally vote")
```

The statements associated with the `else` are executed `if` the condition is false. For this reason the `else` does not need a condition. Note that a colon (`:`) follows the `else` keyword, and that the statements associated with the `else` are *indented*.

In summary:

```
if <condition>:
    statements to execute if condition is true
else:
    statements to execute if the condition is false
```

10.13.2 If...Elif...Else Statement

When we have to decide between more than two scenarios, we need to use an `if...elif...else` statement.

The keyword `elif` stands for `else if` and works in a similar fashion to an `if` statement; it has a condition that needs to evaluate to true for its associated statements to be executed. Note that a colon (`:`) follows the condition, and that the statements associated with the `elif` are *indented*.

Have a look at the following program that evaluates a person's BMI (body mass index). Three scenarios are considered:

1. A person with a BMI of under 18 is considered to be underweight.
2. A person with a BMI of over 25 is considered to be overweight.
3. A person with a BMI between 18 and 25 is considered to have a healthy weight.

Python Source Code

```
#Author: Damir Azhar
#Date: 23/11/15
#Purpose: To evaluate a person's BMI

bmi = int(input("Please enter your BMI: "))
if bmi < 18:
    print("You may be underweight")
elif bmi > 25:
    print("You may be overweight")
else:
    print("Your weight is healthy")
```

Multiple `elif` statements can be used together to decide between several scenarios. The general format is as follows:

```
if <condition 1>:
    statements to execute if condition 1 is true
elif <condition 2>:
    statements to execute if condition 2 is true
                        .
                        .
elif <condition n>:
    statements to execute if condition n is true
else:
    statements to execute if previous n conditions
    are false
```

10.13.3 Comparison operators

The comparison operators are used to compare two different things. They result in a boolean value, so they can be used anywhere that a boolean expression (such as a boolean condition) is required. The following table shows the comparison operators and their meanings.

Operation	Symbol	Example
Less than	<	`age < 18`
Less than or equal to	<=	`age <= 18`
Greater than	>	`age > 18`
Greater than or equal to	>=	`age >= 18`
Equal to	==	`age == 18`
Not equal to	!=	`age != 18`

Note that the equality test (the "equal to" operator) uses two equals signs (==) to distinguish it from the assignment operator which uses a single equals sign (=).

10.13.4 Logical operators

The logical operators are used to combine boolean expressions. They normally create a more complex condition from simpler conditions. The following table specifies the logical operators:

Operation	Symbol	Example
Logical AND	`and`	`(age > 12) and (age < 20)`
Logical OR	`or`	`(age < 5) or (age > 18)`
Logical NOT	`not`	`not (myAge == yourAge)`

These operators are used in a similar way to the logical operators we encountered in spreadsheet formulae.

and

The logical AND operator is used to join two boolean expressions together to form a new, more complex boolean expression. The new boolean expression is true *only* when both of the parts are true. It is false if at least one of the parts are false. For example, the expression:

```
(number > 1) and (number < 10)
```

is only true when both `(number > 1)` is true and `(number < 10)` is true. This occurs only if `number` is between 2 and 9 inclusive.

or

The logical OR operator is used to join two boolean expressions together to form a new, more complex boolean expression. The new boolean expression is true when at least one of the parts is true. It is false only when both parts are false. For example, the expression:

```
(number < 1) or (number > 10)
```

is true either if `(number < 1)` is true, or if `(number > 10)` is true. This occurs when `number` is not between 1 and 10 inclusive.

not

The logical NOT operator is used to reverse the truth value of an existing boolean expression. The new expression is true only if the original expression was false. For example, the expression:

```
not ((number < 1) or (number > 10))
```

is true only when the expression `(number < 1) or (number > 10)` is false. That occurs when `number` is between 1 and 10 inclusive.

10.13.5 Example

The following program is used to make some decisions about a person based on their age.

Python Source Code

```
#Author: Andrew Luxton-Reilly
#Date: 6/05/06
#Purpose: To decide if a person is a teenager or not

age = int(input("Please enter your age: "))
if (age >= 13) and (age <= 19):
    print ("You are a teenager")
    print ("That means you are between 13 and 19")
else:
    print ("You are not a teenager")
```

```
11      print ("That is OK.  I am not either")
12  print ("Goodbye")
```

The following output would be produced if the user entered their age as 19:

```
Please enter your age: 19
You are a teenager
That means you are between 13 and 19
Goodbye
```

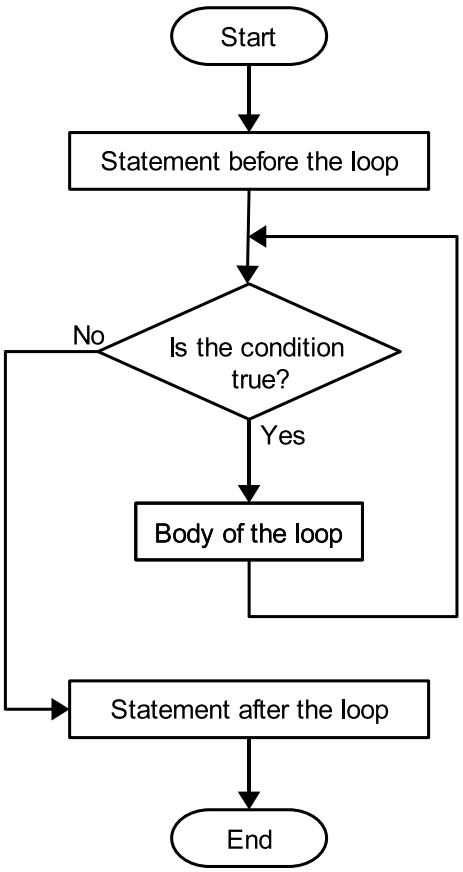
If the user entered their age as 27, the output would be:

```
Please enter your age: 27
You are not a teenager
That is OK.  I am not either
Goodbye
```

10.14 While loops

A `while` loop is a programming construct used to repeatedly execute a number of statements. We use a while loop to test a given condition, and if the condition is true, then we execute a set of statements (much like an `if` statement). However, in a while loop, once we have finished executing the statements, we check the condition again. If the condition is still true, then we execute the statements again. This process continues until the condition is false.

The following flow diagram illustrates how a `while` loop works in a program.



The formal syntax for a while loop is as follows:

```
while <condition>:
    statements to execute when the condition is true
```

There are some important notes to remember about a `while` loop:

- The `<condition>` must be a boolean expression.
- There is a colon (`:`) after the condition.
- The `statements to execute when the condition is true` are a block of statements. These statements are executed if the condition is true.
- All the statements in the block must be uniformly indented (i.e. they should all be indented using the same number of spaces)
- After the statements in the block have finished executing, the condition is checked again and if it is still true then the statements in the block will be executed again. This process continues until the condition is false.

For example, the following program:

Python Source Code

```
1 start = 7
2 end = 12
3 counter = start
4
5 while counter <= end:
```

```
6       print (counter)
7       counter = counter + 1
```

will produce the output:

```
7
8
9
10
11
12
```

10.14.1 Example

Imagine that we want a program to print out the 7 times table. We could write the program using a series of print statements as follows:

Python Source Code

```
1  print ("7 * 1 = 7")
2  print ("7 * 2 = 14")
3  print ("7 * 3 = 21")
4  print ("7 * 4 = 28")
5  print ("7 * 5 = 35")
6  print ("7 * 6 = 42")
7  print ("7 * 7 = 49")
8  print ("7 * 8 = 56")
9  print ("7 * 9 = 63")
10 print ("7 * 10 = 70")
```

However, if we write the program this way, we have to do all the calculations ourselves. It would be very time consuming to change the program to print out the 8 times table. A better version would use a variable to store the information about the times tables we are calculating. We could write the program as follows:

Python Source Code

```
1  number = 7
2  print (number, "* 1 =", number * 1)
3  print (number, "* 2 =", number * 2)
4  print (number, "* 3 =", number * 3)
5  print (number, "* 4 =", number * 4)
6  print (number, "* 5 =", number * 5)
7  print (number, "* 6 =", number * 6)
8  print (number, "* 7 =", number * 7)
9  print (number, "* 8 =", number * 8)
10 print (number, "* 9 =", number * 9)
11 print (number, "* 10 =", number * 10)
```

Although this is certainly better than the previous version, we can recognize that we are repeating very similar statements in this case. We can use a loop to make the code much more efficient. Furthermore, using a loop will make it easier to extend the times table beyond 10 if we wish. We could write a better version of this program as follows:

Python Source Code

```
number = 7
start = 1
end = 10

count = start
while (count <= end):
        print (number, "*", count, "=", number * count)
        count = count + 1
```

This program allows us to easily change the times table that will be printed by changing the value used for the `number`, the `start` or the `end` values. The loop will print the times table for the number given, starting with the `start` number and finishing with the `end` number.

10.15 Turtle Graphics

The term **turtle graphics** describes the process of creating line graphics using a cursor (the "turtle"), along with commands for movement and drawing. Turtle graphics started as an extension to the Logo programming language created by Seymour Papert and Wally Feurzeig in 1967. Logo was designed to be an educational programming language, so it had a simple syntax that children were supposed to be able to master rapidly. Turtle graphics was a popular tool for teaching programming as it allowed the user to:

- see how their program behaved directly on screen,
- compare this behaviour to what they had intended, and
- identify where any problems occured and correct them.

In the following section we will look at using turtle graphics with the Python programming language.

10.15.1 Importing Python Modules

Before we look at turtle graphics, we need to be able to import Python modules. Like many programming languages, Python distinguishes between two forms of commands:

1. Ones built into the core language that are always available like `print()` and `input()`.
2. Ones defined in modules (libraries) that you need to load before you use.

A programmer can define their own modules or use existing ones that are available to all Python users. The `turtle` module is an example of the second type. You need to load the `turtle` module before you can use turtle graphics. You can load any module into

the IDLE environment using the `import` command. The following line of code would be used to import the `turtle` module:

Python Source Code

```
import turtle
```

When developing a program that requires commands found in modules, importing these modules is usually a good first step.

Once you have imported a module, you can use the commands defined in it. To do this the following syntax is used:

```
nameOfModule.nameOfCommand()
```

You will see several examples in the next section where we look at basic turtle commands.

10.15.2 Basic Turtle Commands

With turtle graphics, you can think of the turtle (i.e. the cursor) as a pen. By default the tip of the pen is down so when the turtle moves a line is drawn. If the tip is up, the turtle can be moved without drawing a line. The following two commands let you specify whether you want the tip to be up or down:

`turtle.penup()`

- Lifts the pen so that turtle movement *does not* draw a line.

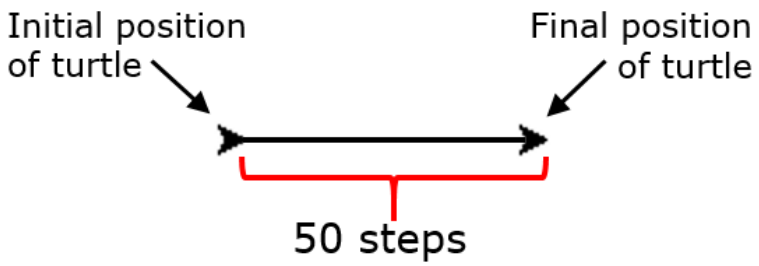
`turtle.pendown()`

- Lowers the pen so that turtle movement *does* draw a line.

The next set of commands are used to move the turtle by a specified amount:

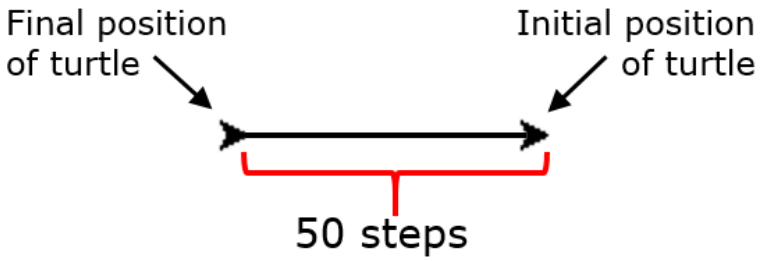
`turtle.forward(n)`

- Moves the turtle forward in the direction it is facing by n steps.
- For example `turtle.forward(50)` moves the turtle forward by 50 steps.



`turtle.backward(n)`

- Moves the turtle backward from its facing direction by n steps.
- For example `turtle.backward(50)` moves the turtle backward by 50 steps.

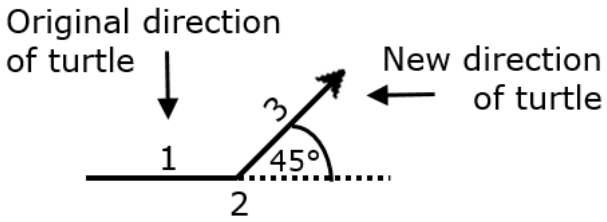


The next two commands are used to turn the turtle by a specified angle. Note that the turtle rotates using its tip as a pivot.

`turtle.left(n)`

- Turns the turtle n degrees counter-clockwise.
- For example:
  ```
  turtle.forward(25)
  turtle.left(45)
  turtle.forward(25)
  ```

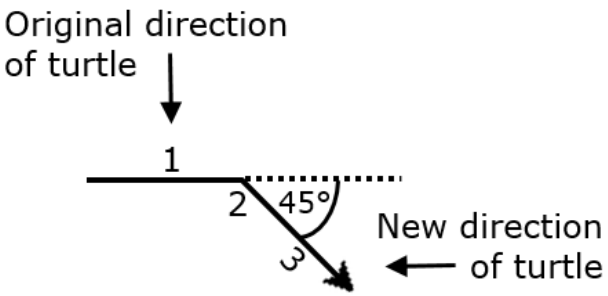
1. Moves the turtle forward 25 steps,
2. Turns the turtle 45 degrees counter-clockwise, and then
3. Moves the turtle 25 steps in the new direction it is facing.



`turtle.right(n)`

- Turns the turtle n degrees clockwise.
- For example:
  ```
  turtle.forward(25)
  turtle.right(45)
  turtle.forward(25)
  ```

1. Moves the turtle forward 25 steps,
2. Turns the turtle 45 degrees clockwise, and then
3. Moves the turtle 25 steps in the new direction it is facing.



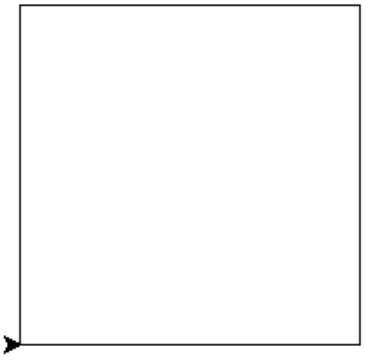
10.15.3 Example - Drawing A Square

In this example we will look at using turtle graphics to draw a square. A square is a polygon with four sides, where the sides are of equal length, and the angle between sides is 90 degrees. The following Python code will draw a square with sides that are 200 steps in length:

Python Source Code

```
import turtle
turtle.forward(200)
turtle.left(90)
turtle.forward(200)
turtle.left(90)
turtle.forward(200)
turtle.left(90)
turtle.forward(200)
turtle.left(90)
```

The resulting square is shown below. Note the position and orientation of the turtle at the end of the drawing process.



10.15.4 Example - Using A While Loop For Drawing

The exact same square can be drawn with a while loop instead. In the previous example, each side of the square was drawn using the following two lines of code:

```
turtle.forward(200)
turtle.left(90)
```

In order to draw the square we would place these two lines of code within a while loop as shown on the next page:

Python Source Code

```
1 import turtle
2
3 count = 0
4 while count < 4:
5     turtle.forward(200)
6     turtle.left(90)
7     count = count + 1
```

CHAPTER 11

LaTeX

11.1 Introduction

LaTeX is a document preparation system. It is not a word processor, it does not check spelling, and it is not a WYSIWYG program. LaTeX is designed for typesetting. It allows a user to specify the way a document will look when it is printed, based on the structure of that document. In short, LaTeX is used to *format* a text document.

LaTeX is a program that is available for most common platforms (e.g. Macintosh OS, Windows, Linux). It takes an input file that consists of plain ASCII text (much like an HTML source file) and produces output that is in a format such as device independent (dvi), postscript (ps), or portable document format (pdf). This output will be displayed exactly as it will appear when printed.

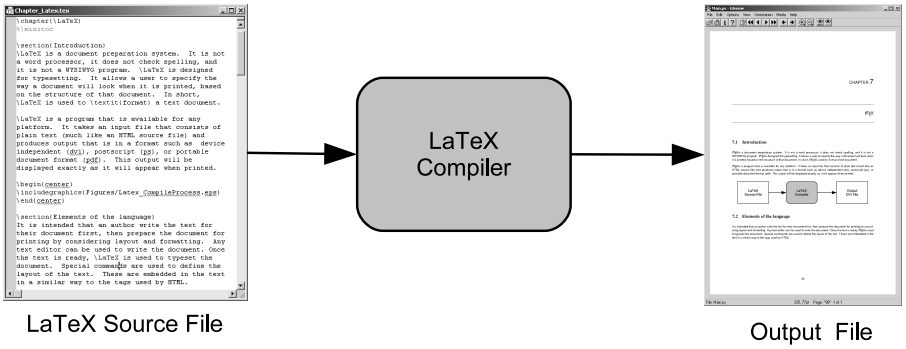


Figure 11.1: Running LaTeX

Note: The input file (the LaTeX source file) should use a filename that ends with the file extension `.tex` to identify it as a LaTeX file.

11.1.1 Why would we use LaTeX?

The LaTeX program was originally developed to help mathematicians typeset their documents. It is still the best product today to create professional documents that use mathematical equations. It is also used in many academic institutions to typeset theses and other large documents.

Some of the commonly cited advantages of LaTeX are:

- The finished documents look as if they were professionally printed.
- LaTeX supports typesetting mathematical equations in a very convenient way.
- Long documents can be easily managed
- The table of contents, indexes, glossaries and bibliographies can be generated easily.
- LaTeX is free and runs on a wide range of platforms.

In short, if you are creating a very long document (such as a thesis or a book), or if you really care about the way your documents look, then LaTeX is highly recommended. If you are only writing short documents, and you don't care about the way they look when printed, then a simple word processor such as Microsoft Word will probably fulfil your needs.

11.2 Overview of the language elements

It is intended that an author write the text for their document first, then prepare the document for printing by considering layout and formatting. Any text editor can be used to write the document. Once the text is ready, LaTeX is used to typeset the document. Special commands are used to define the layout of the text. These are embedded in the text in a similar way to the tags used by HTML.

11.2.1 Comments

A comment is used in LaTeX for the same purpose as in XHTML or Python. A comment provides information for human beings who are reading the source code. The computer completely ignores all comments.

The symbol for a comment in LaTeX is the percentage sign (`%`). All text that appears to the right of a percentage sign is ignored, until the end of the line is reached. For example, the following code is valid in LaTeX , but will not result in any output, since all the text is composed of comments.

LaTeXSource Code

```
%Author: Andrew Luxton-Reilly
%Date: 09/05/2006
```

```
%A simple introduction to LaTeX, written for
%COMPSCI 111 students.
```

11.2.2 Whitespace

"Whitespace" characters such as Space or Tab leave a blank space in the document. LaTeX will contract several consecutive spaces in the document to a single space. It will also ignore "whitespace" at the start of a line, and a single line break (i.e. generated by pressing the Enter key once) is treated as a single "space".

For example, the following code:

LaTeX Source Code

```
All of        these
                   words
       will appear in
a single sentence.
```

will produce the output:

All of these words will appear in a single sentence.

11.2.3 Commands

Commands in LaTeX all start with a backslash character (\), followed by a name that consists only of letters. They use the form:

```
\commandname
```

For example, the command:

LaTeX Source Code

```
\newpage
```

will end the current page, and any following text will appear on a new page.

Some LaTeX commands use the form:

```
\commandname{argument}
```

These commands apply to the **argument** enclosed in the curly braces {} that immediately follow the name of the command. For example, the command:

LaTeX Source Code

```
\section{Computer Science}
```

will create a new section entitled “Computer Science”.

Still other LaTeX commands use the form:

`\commandname[`**options**`]{`**argument**`}`

Commands of this sort require an argument to be enclosed in curly braces, but they also have various options that may affect the way the command is applied. The options are enclosed in square brackets `[]`, and are included between the command name and any curly braces that might follow. For example, the command:

LaTeXSource Code

```
\section[Computer Science]{Introduction to the Science
of Computing}
```

will create a new section entitled “Introduction to the Science of Computing”, however, in the table of contents, this section will be referred to as “Computer Science”.

Note the following:

- Commands in LaTeX are case sensitive. The command `\Large` is different than the command `\large`.
- If a command uses options, then it is not compulsory to include them.

11.2.4 Environments

Formatting in a LaTeX document is controlled by both commands, and environments. A command is normally used to apply a simple change. Commands are often applied within a paragraph of text. When we change the environment, we always start a new paragraph. As such, environments are normally applied to larger groups of text (such as paragraphs, pages, or entire documents).

The format to change the environment is as follows:

`\begin{`**environment name**`}`
Text enclosed by the environment goes here
`\end{`**environment name**`}`

For example, to centre some text on the page, we could use the `center` environment as follows:

LaTeXSource Code

```
\begin{center}
This text is centred.
\end{center}
```

The source code above would produce the following output:

This text is centred.

Note the spelling of the word `center` uses American spelling.

11.2.5 Special characters

There are 10 keyboard characters that have a special meaning in LaTeX . You cannot simply type them directly into a LaTeX document. Instead, you must use special commands to get them to appear on the page.

Description	Character	Special command to make them appear on the page
Backslash	\	$\backslash$
Dollar sign	$	\$
Percent sign	%	\%
Caret	^	\^
Ampersand	&	\&
Underscore	_	_
Tilde	~	\~
Hash	#	\#
Open curly brace	{	$\{$
Close curly brace	}	$\}$

11.2.6 Paragraphs and line breaks

Paragraphs are separated by leaving a blank line in the input. Remember that a single line break (caused by pressing the Enter key once) will be ignored by LaTeX , but pressing the Enter key twice will leave a blank line which will be interpreted as starting a new paragraph. For example, the following LaTeX code:

LaTeX Source Code

```
Each paragraph should contain only one single idea.
If you find that a single paragraph contains more
than one idea, then you should split that paragraph.
If you find that two different paragraphs are about
the same idea, then those paragraphs should be joined
together.

The normal way to signify the start of a new paragraph
is to indent the first line of the paragraph.
```

would be displayed as:

Each paragraph should contain only one single idea. If you find that a single paragraph contains more than one idea, then you should split that paragraph. If you find that two different paragraphs are about the same idea, then those paragraphs should be joined together.

The normal way to signify the start of a new paragraph is to indent the first line of the paragraph.

A line break can be inserted into the output using the command `\\`. There are occasions where this is useful. For example, the code:

LaTeXSource Code

```
\begin{center}
The long night passes,\\
the hairy caterpillar lies\\
in the new-born day\\
\end{center}
```

would be displayed as:

The long night passes,
the hairy caterpillar lies
in the new-born day

11.3 Document class

The structure of a LaTeX document is always the same. First, we need to specify the kind of document we are creating. Then we use `\begin{document}` and `\end{document}` to surround the entire document that we want to typeset.

A simple LaTeX file would look like:

`\documentclass[`**options**`]{`**documenttype**`}`

`\begin{document}`

... document to be formatted goes here

`\end{document}`

11.3.1 Classes of document

You can use LaTeX to create a number of different kinds of document. You can think of the document classes as being like templates. They define the overall appearance and style of your document.

The format for this command is:

`\documentclass[`**options**`]{`**documenttype**`}`

There are four different default document types that come with LaTeX . Although it is possible to define your own document type, it is a difficult process that is unnecessary for most people. It is worth noting that many institutions have created document classes that define the standard formatting for documents published at that institution (e.g. many universities have created a document class for publishing theses). The four standard document styles are described below:

`article` Designed for shorter documents, essays, and articles for publication. The article document class does not have chapters, and the title appears at the top of the first page, rather than having a title page of its own.

`report` A report is used for longer documents. Reports can have different chapters and the title appears on a page by itself.

`book` Books are formatted in a similar way to reports, except that it is assumed that a book will be printed on both sides of the paper, so the layout and margins are adjusted accordingly.

`letter` This document class is designed for producing personal letters.

The options that can be used are as follows:

`10pt, 11 pt, 12 pt` Used to define the normal size for text that appears in the book. All other fonts used for headings, footnotes etc. will be scaled accordingly. If no option is specified, then 10pt will be used.

`a4paper, letterpaper` Defines the size of the page. The default is `letterpaper`.

`draft` Creates a draft version. LaTeX will indicate problems with the layout (that it can identify) by placing a small square in the right-hand margin.

`onecolumn, twocolumn` Specifies whether to use page layout that consists of one or two columns.

`oneside, twoside` Used only for `book` and `report` types. Formats the document for one-sided, or two-sided printing. By default, `article` and `report` are formatted for one-sided printing, but `book` is formatted for two-sided printing.

`titlepage, notitlepage` Specifies whether to start a new page after displaying the title. By default, `report` and `book` will start a new page, but `article` will not.

For example, this chapter that you are reading now is formatted using LaTeX and uses the document class declaration

LaTeXSource Code

```
1 \documentclass[a4paper,twosided]{book}
```

11.3.2 Preamble

The area of a LaTeX source file between the `\documentclass` declaration and the `\begin{document}` command is called the "preamble". This area is used to put any commands that will alter the way the document is typeset. Later we will look at the kinds of things that could appear in the preamble, but at the moment, that space should remain empty.

11.3.3 A simple LaTeX document

The following example shows a very simple LaTeX document. The following code:

LaTeX Source Code

```
\documentclass{article}

%This area is the preamble.  We do not need
%any commands here

\begin{document}
A simple document.
\end{document}
```

would produce an entire page containing only the text:

A simple document.

11.4 Titles

The first thing that you would normally put inside the `document` environment is the title of your document. This would be followed by the author's name and the date. Although these things are not *required*, they are usually included. The following table summarizes the commands used to create the title page for our document.

Command	Purpose
`\title{`**title name**`}`	Defines the title
`\author{`**author name**`}`	Defines the author
`\date{`**date**`}`	Defines the date (optional)
`\maketitle`	Displays the information

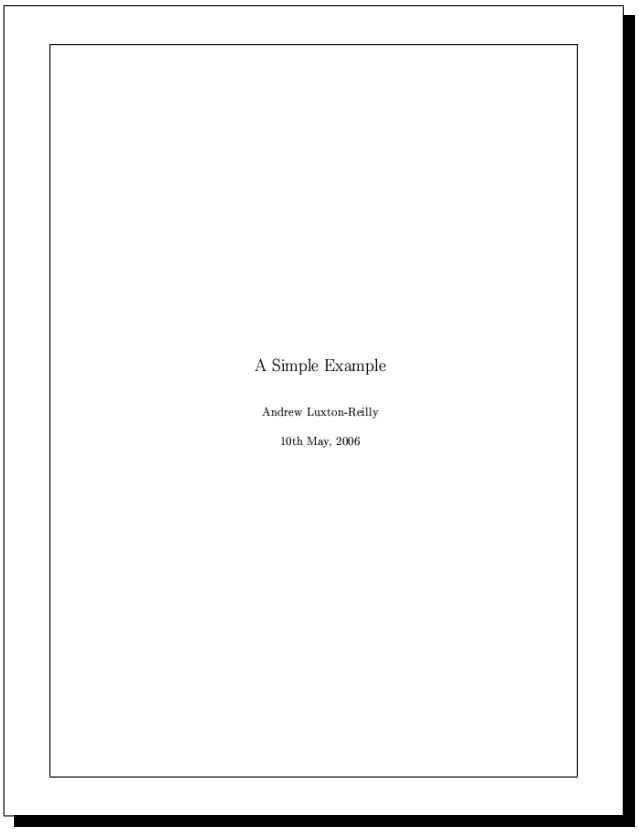
If the **date** is omitted, then the current date will be used on the title page. This can be useful to keep track of the document version if it is regularly being revised.

The `\maketitle` command is important. It must be used *after* the title, author and date have been defined, and it is used to format and display the information. For example, the following code:

LaTeX Source Code

```
\documentclass{report}

\begin{document}
\title{A simple example}
\author{Andrew Luxton-Reilly}
\date{10th May, 2006}
\maketitle

This document contains a simple example
\end{document}
```

would result in the following title page:



11.5 Structuring a document

One of the advantages of LaTeX is that it encourages you to concentrate on the structure of the document. You can divide your document up into different chapters, sections and subsections using special commands. These commands are described in this section.

11.5.1 Parts

If you want to split your document up into parts, without interfering with the chapter or section numbering, then you can use the `\part` command. The formal syntax for this is:

`\part{`**part name**`}`

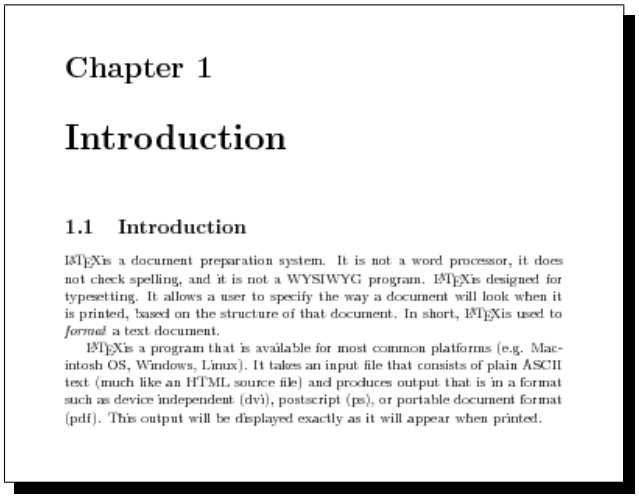
The name of the part will be used in the table of contents, and will also be used on a page that separates the different parts of your document. The `\part` command is not often used, except in very large books.

11.5.2 Chapters

The `\chapter` command can only be used if your document class is `report` or `book`. An `article` cannot be divided up into different chapters. The syntax for this command is:

`\chapter{`**chapter name**`}`

The name of the chapter will be used in the table of contents. The `chapter` command will start a new chapter which will start at the top of a new page. The name of the chapter will be formatted and displayed in a large font size at the top of the page, and all section numbering will be reset.


Chapter 1

Introduction

1.1 Introduction

LaTeXis a document preparation system. It is not a word processor, it does not check spelling, and it is not a WYSIWYG program. LaTeXis designed for typesetting. It allows a user to specify the way a document will look when it is printed, based on the structure of that document. In short, LaTeXis used to *format* a text document.

LaTeXis a program that is available for most common platforms (e.g. Macintosh OS, Windows, Linux). It takes an input file that consists of plain ASCII text (much like an HTML source file) and produces output that is in a format such as device independent (dvi), postscript (ps), or portable document format (pdf). This output will be displayed exactly as it will appear when printed.

11.5.3 Sections and subsections

A number of section commands are available. These can be used in the `article`, `report` or `book` classes. The sections are numbered and the name of the section will be displayed in a large bold font. The following commands can be used:

`\section{`**section name**`}`

`\subsection{`**subsection name**`}`

```
\subsubsection{subsubsection name}
```

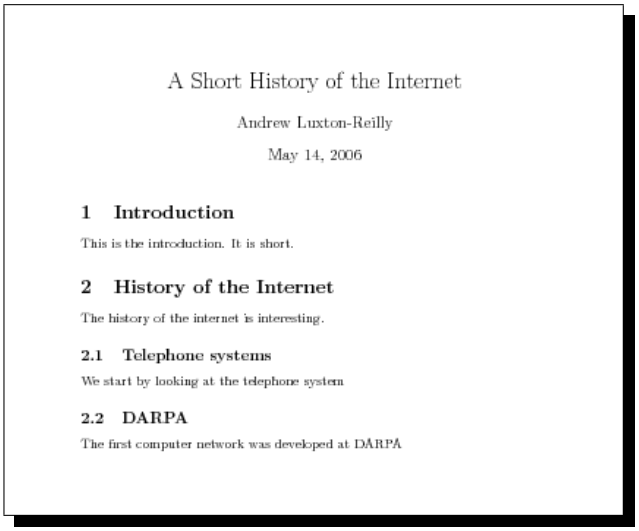
Note that the `subsubsection` name will not be numbered by default, nor will it appear in the table of contents.

For example, the following page has been divided up using sections and subsections:

LaTeX Source Code

```
\documentclass{article}

\begin{document}

\title{A Short History of the Internet}
\author{Andrew Luxton-Reilly}
\maketitle

\section{Introduction}
This is the introduction.  It is short.

\section{History of the Internet}
The history of the internet is interesting.

\subsection{Telephone systems}
We start by looking at the telephone system

\subsection{DARPA}
The first computer network was developed at DARPA

\end{document}
```

When the page is displayed, it will look like the following:



A Short History of the Internet

Andrew Luxton-Reilly

May 14, 2006

1 Introduction

This is the introduction. It is short.

2 History of the Internet

The history of the internet is interesting.

2.1 Telephone systems

We start by looking at the telephone system

2.2 DARPA

The first computer network was developed at DARPA

11.6 Table of contents

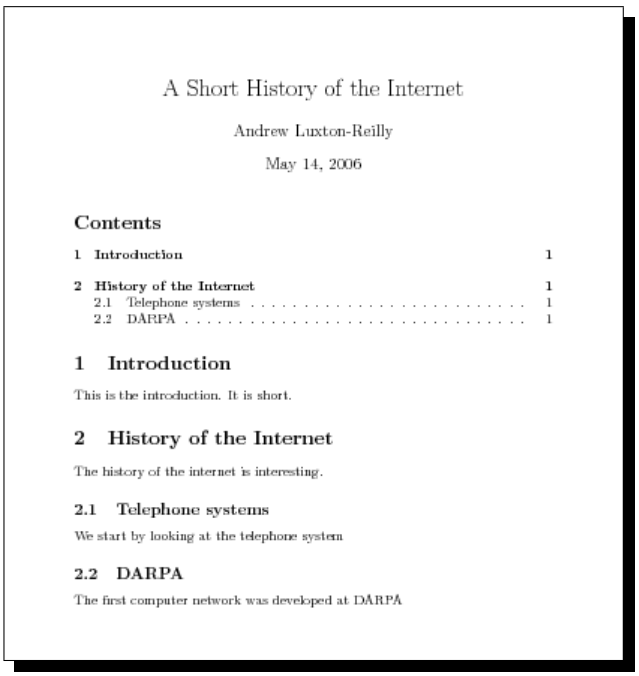
A table of contents can be inserted using the command `tableofcontents`. By default, parts, chapters, sections and subsections will be included in the table of contents. The page numbers used in the table of contents will be automatically generated. It is worth noting that you have to run the input file through LaTeX twice in a row in order to get the table of contents to work properly (the first time does all the layout of the pages, then the second time inserts the correct page numbers into the table of contents).

For example, the table of contents produced by the following document:

LaTeXSource Code

```
\documentclass[a4paper]{article}

\begin{document}

\title{A Short History of the Internet}
\author{Andrew Luxton-Reilly}
\maketitle
\tableofcontents

\section{Introduction}
This is the introduction.  It is short.

\section{History of the Internet}
The history of the internet is interesting.

\subsection{Telephone systems}
We start by looking at the telephone system

\subsection{DARPA}
The first computer network was developed at DARPA

\end{document}
```

would look like the following:



A Short History of the Internet

Andrew Luxton-Reilly

May 14, 2006

Contents

1 Introduction

This is the introduction. It is short.

2 History of the Internet

The history of the internet is interesting.

2.1 Telephone systems

We start by looking at the telephone system

2.2 DARPA

The first computer network was developed at DARPA

11.7 Footnotes

Footnotes can easily be inserted using LaTeX. The command to insert a footnote is:

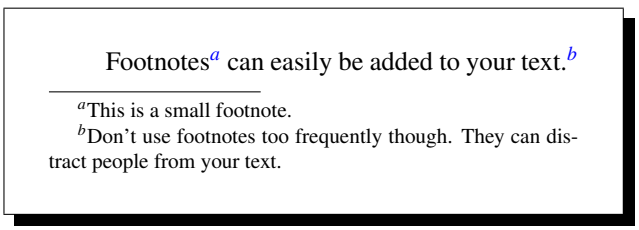
```
\footnote{footnote text}
```

The footnote will always appear at the bottom of the current page. For example, the code:

LaTeX Source Code

```
1 Footnotes\footnote{This is a small footnote.} can
2 easily be added to your text.\footnote{Don't use
3 footnotes too frequently though.  They can distract
4 people from your text.}
```

will be displayed as:



Footnotes[a] can easily be added to your text.[b]

[a] This is a small footnote.

[b] Don't use footnotes too frequently though. They can distract people from your text.

11.8 Symbols used in text

When you enter plain text, there are frequently symbols that you want to use, but you find that they are not available in plain ASCII text. These symbols can be included in the final output if you know how to represent them using LaTeX.

11.8.1 Quote marks

" The quote marks that are normally typed from the keyboard (") are called unidirectional quotes. They are not used in LaTeX. Instead, we create the more professional looking directional quotes. For single quote marks, we use the single quote (`) and apostrophe('). For double quote marks, we use the same characters, but use two characters instead of just one. For example, the following code:

LaTeXSource Code

```
1 She said, ``That's very interesting''.
```

would be displayed as:

She said, “That’s very interesting”.

In some of the editors used to write LaTeX code (such as TeXnicCenter), the options can be set to automatically replace the " with either `` or '' depending on whether a letter or a space appears before you press the " key. In other words, it will insert the LaTeX codes for the appropriate quote marks when you type the key for ".

11.8.2 Special symbols

A range of different commands are used to represent accents, ligatures and other symbols common in written English. The most commonly used symbols are summarised in the following table.

Description	Example	LaTeX command
Acute (fada)	é	\'{e}
Grave	è	\`{e}
Circumflex	ê	\^{e}
Umlaut	ë	\"{e}
Tilde	ñ	\~{n}
Macron	ō	\={o}
Bar-under	o̲	\b{o}
Dot-over (séımȯú)	ṁ	\.{m}
Dot-under	ṣ	\d{s}
Breve	ŭ	\u{u}
Háček (caron)	č	\v{c}
Long umlaut	ő	\H{o}
Cedilla	ç	\c{c}
O-E ligature	œ, Œ	\oe, \OE
A-E ligature	æ, Æ	\ae, \AE
A-ring	å, Å	\aa, \AA
O-slash	ø, Ø	\o, \O

11.8.3 Dashes

A single dash is used to represent a hyphen. For example, the code:

LaTeX Source Code

```
merry-go-round
```

will appear as:

merry-go-round

However, in some circumstances (such as when we are discussing a range of pages), we want the dash to be longer (called an en dash in typesetting). In this case, we use two dashes in the LaTeX code. For example, the code:

LaTeX Source Code

```
pages 34--45
```

will appear as:

pages 34–45

Finally, there are times that we want to use an extra long dash (called an em dash). In this case, we use three subsequent dashes in the LaTeX code. For example, the code:

LaTeX Source Code

```
We can use an em dash like parentheses---that is,
to enclose a phrase---or they can be used to separate
```

```
the end of a sentence from the main text.
```

will appear as:

> We can use an em dash like parentheses—that is, to enclose a phrase—or they can be used to separate the end of a sentence from the main text.

11.8.4 Ellipsis

Ellipsis is the omission of a word or phrase from some quoted text. It is usually represented using three dots. We cannot simply put three full stops in a row, since the spacing generated for three full stops will be incorrect. We use the LaTeX command `\ldots` to insert ellipsis into our text. For example, the code:

LaTeX Source Code

```
A well-known speech begins with the phrase ``Four
score and seven years ago \ldots''.
```

will be displayed as:

> A well-known speech begins with the phrase "Four score and seven years ago . . . ".

11.8.5 Spaces

Spaces between words can be stretched or compressed to ensure that the finished text looks good on the page. When LaTeX is deciding where to put the end of a line, it tries to split the lines at a space. If it cannot find a suitable space, then it will automatically hyphenate a word to fit the text.

Unbreakable spaces

There are some places where it is inappropriate to split a line. For example, we would not normally want to split a person's initials from their surname (i.e. in the case of E. E. Smith, we would not want to split the line with the initials at the end of one line and the surname on the next line).

To ensure that LaTeX does not split the line at the specified point, we can use the tilde character (`~`) instead of a space. This will tell LaTeX to use a normal width space, but the line is not permitted to be broken at that point. We would normally write this as `E.~E.~Smith`.

Normal width spaces after a full stop

When text is typeset, it is conventional to leave a larger gap between different sentences than between words. If we were typing with a typewriter, then we would normally leave two spaces after typing a full stop at the end of a sentence. LaTeX will automatically leave a larger gap after a full stop, so we do not have to manually tell it to do so.

However, this can cause problems when we use a full stop in abbreviations, since LaTeX thinks we have reached the end of a line. For example, the use of the full stop in the following text results in incorrect spacing:

> E.g. when we use the full stop after a title such as Mr. Jones, then LaTeX thinks we are ending a sentence. It will leave a space the same size after a title as it does after a sentence, i.e. the space will be incorrect.

We can correct this problem by using a tilde (~) as discussed previous. This will also tell LaTeX not to break the line at that point. If we are happy for the line to be broken, but we want a normal sized space, then we can use a backslash followed by a space (`\ `). This will tell LaTeX to leave a normal sized space and allow the line to be broken at that point. The following text has been generated using the backslash and space command to ensure the spaces are the correct size. Compare this version with the previous one and look closely at the different amounts of space that occur after the full stop.

> E.g. when we use the full stop after a title such as Mr. Jones, then LaTeX thinks we are ending a sentence. It will leave a space the same size after a title as it does after a sentence, i.e. the space will be incorrect.

The LaTeX source code for the text above would be:

LaTeX Source Code

```
E.g.\ when we use the full stop after a title
such as Mr.\ Jones, then \lx thinks we are
ending a sentence.  It will leave a space the
same size after a title as it does after a
sentence, i.e.\ the space will be incorrect.
```

11.9 Text styles

There are a range of commands that can be used to change the style of the font that is used. Note that it is usually better to allow LaTeX to decide what font style to apply

based on the structure of the document, but there are times that it is useful to manually change the font style.

11.9.1 Emphasis

One of the most common ways that we want to change the style of text is to emphasise a word or phrase. The command to do this is:

`\emph{`**argument**`}`

For example, the source code:

LaTeXSource Code

```
Text should only be emphasised \emph{rarely}.
```

will appear as:

Text should only be emphasised *rarely*.

11.9.2 Font styles

The appearance of the font can be changed using a range of different commands. They are included here, but their use is not recommended in a normal document.

Command	Meaning	Example
`\textbf{`**argument**`}`	Bold	A short **example**
`\textit{`**argument**`}`	Italic	A short *example*
`\textsl{`**argument**`}`	Slanted	A short *example*
`\textsf{`**argument**`}`	Sans-serif	A short example
`\textrm{`**argument**`}`	Serif	A short example
`\texttt{`**argument**`}`	Monospace	A short `example`
`\textsc{`**argument**`}`	Small Capitals	A short EXAMPLE

11.9.3 Font size

The size of the font can be set to a range of different sizes. It is not normally necessary to manually set the size of fonts, but the commands to set the font size are included in the table below:

Command	Example
\tiny	A small example
\scriptsize	A small example
\footnotesize	A small example
\small	A small example
\normalsize	A small example
\large	A small example
\Large	A small example
\LARGE	A small example
\huge	A small example
\Huge	A small example

These commands will change the font used for any text that appears from that point forward. For example, if you set the text to be \huge, then all text that appears after the command will be huge. To limit the scope of the command, you can use curly braces. However, the curly braces are used in a different way than we have seen before. The curly braces enclose all the text that we want to change, and the command to change the size occurs *inside* the curly braces. The format for this is:

{**size command** text that you want to define the size of}

For example, the following code:

LaTeX Source Code

```
1 This text ranges in size from {\tiny very, very}
2 {\footnotesize small} to {\Large  very, very}
3 {\Huge large.}
```

will be displayed as:

This text ranges in size from very, very small to very, very large.

11.10 Alignment environments

We can align text to the left or right margin, centre the text, or justify the text. To change the alignment, we use one of the environments described in this section.

11.10.1 Left aligned text

The flushleft environment will align all the text in the paragraph to the left side. For example, the code:

LaTeXSource Code

```
\begin{flushleft}
The flushleft environment will cause the text to be
aligned to the left side of the page.

Short sentences that end with a line break\\
will also be aligned\\
to the left.\\
\end{flushleft}
```

will be displayed as:

The flushleft environment will cause the text to be aligned to the left side of the page.
Short sentences that end with a line break
will also be aligned
to the left.

11.10.2 Right aligned text

The `flushright` environment will align all the text in the paragraph to the right side. For example, the code:

LaTeXSource Code

```
\begin{flushright}
The flushright environment will cause the text to be
aligned to the right side of the page.

Short sentences that end with a line break\\
will also be aligned\\
to the right.\\
\end{flushright}
```

will be displayed as:

The flushright environment will cause the text to be aligned to the right side of the page.
Short sentences that end with a line break
will also be aligned
to the right.

11.10.3 Centred text

The `center` environment will align all the text in the paragraph to the centre. For example, the code:

LaTeX Source Code

```
\begin{center}
The center environment will cause the text to be
aligned to the centre of the page.

Short sentences that end with a line break\\
will also be aligned\\
to the centre.\\
\end{center}
```

will be displayed as:

The center environment will cause the text to be
aligned to the centre of the page.
Short sentences that end with a line break
will also be aligned
to the centre.

11.11 List environments

Three different list environments are supported by LaTeX. These are similar to those that are used in XHTML, namely, an unordered list, an ordered list and a description list. Each of these list environments are described in turn in this section.

11.11.1 Unordered lists

The `itemize` environment is used to create a list that consists of bullet points. The format for the list is as follows:

```
\begin{itemize}
\item text that makes up a bullet point
\end{itemize}
```

For example, the following LaTeX code:

LaTeX Source Code

```
There are three kinds of lists that are supported.
They are:
\begin{itemize}
\item Itemize
\item Enumerate
```

```
\item Description
\end{itemize}
```

will be displayed as:

There are three kinds of lists that are supported. They are:

- Itemize
- Enumerate
- Description

11.11.2 Ordered lists

The `enumerate` environment is used to create a list that consists of numbered entries. The format for the list is as follows:

```
\begin{enumerate}
\item text that makes up an item in the list
\end{enumerate}
```

For example, the following LaTeX code:

LaTeX Source Code

```
There are three kinds of lists that are supported.
They are:
\begin{enumerate}
\item Itemize
\item Enumerate
\item Description
\end{enumerate}
```

will be displayed as:

There are three kinds of lists that are supported. They are:

1. Itemize
2. Enumerate
3. Description

11.11.3 Description lists

The `description` environment is used to create a list that is used to define (or provide a description of) entries. The format for the list is as follows:

```
\begin{description}
\item[term] description of the term
\end{description}
```

For example, the following LaTeX code:

LaTeX Source Code

```
There are three kinds of lists that are supported.
They are:
\begin{description}
\item[Itemize] lists that use bullet points
\item[Enumerate] lists that are enumerated
\item[Description] lists that describe terms
\end{description}
```

will be displayed as:

There are three kinds of lists that are supported. They are:
Itemize lists that use bullet points
Enumerate lists that are enumerated
Description lists that describe terms

11.12 Quote and quotation environments

Two kinds of environments are used with quotations. They are similar in that they both indent the margins of the quote and separate it from the main body.

11.12.1 Quote

The `quote` environment causes the right and left margins to be indented. Leaving a blank line in a `quote` environment will create a new paragraph. The text at the start of a paragraph will not be indented. This environment is designed for short quotes (hence the lack of indentation at the start of the first line).

For example, the code:

LaTeX Source Code

```
The following quote is one of my favourites from
Groucho Marx:
\begin{quote}
Those are my principles, if you don't like them,
I have others!
\end{quote}
```

will be displayed as:

> The following quote is one of my favourites from Groucho Marx:
>
> Those are my principles, if you don't like them, I have others!

11.12.2 Quotation

The `quotation` environment is very similar to the `quote` environment. It causes the right and left margins to be indented. Leaving a blank line in a `quotation` environment will create a new paragraph. The main difference between `quote` and `quotation` is that the first line of the paragraph is indented in a `quotation` environment. This environment is designed for longer quotes, especially those that have multiple paragraphs.

For example, the code:

LaTeXSource Code

```
The author Neal Stephenson wrote a very interesting
history of operating systems.  Here is a small
extract:
\begin{quotation}
The analogy between cars and operating systems is
not half bad, and so let me run with it for a moment,
as a way of giving an executive summary of our
situation today.

Imagine a crossroads where four competing auto
dealerships are situated. One of them (Microsoft)
is much, much bigger \ldots

There was a competing bicycle dealership next door
(Apple) that one day began selling motorized
vehicles---expensive but attractively styled cars
with their innards hermetically sealed, so that
how they worked was  something of a mystery.
\end{quotation}
```

will be displayed as:

The author Neal Stephenson wrote a very interesting history of operating systems. Here is a small extract:

> The analogy between cars and operating systems is not half bad, and so let me run with it for a moment, as a way of giving an executive summary of our situation today.
>
> Imagine a crossroads where four competing auto dealerships are situated. One of them (Microsoft) is much, much bigger ...
>
> There was a competing bicycle dealership next door (Apple) that one day began selling motorized vehicles—expensive but attractively styled cars with their innards hermetically sealed, so that how they worked was something of a mystery.

11.13 Verbatim environment

Occasionally, we need to print out some text without it being typeset at all. We want the text to appear *exactly* as we type it. This often occurs when we are writing documents about spreadsheet formulae, XHTML, programming code, or even LaTeX itself. In these situations, we use an environment called `verbatim`.

In the `verbatim` environment, every space will be printed exactly as it appears in the source code. Every character will be included exactly as it appears in the source, so we can type special characters or LaTeX commands and they will be reproduced in the output.

For example, the following code:

LaTeX Source Code

```
This    is    not     verbatim text.
\begin{verbatim}
This    is    some    verbatim text.

Normally, using \\ will create a line
break, but not inside
a verbatim environment.
\end{verbatim}
```

will be displayed as:

This is not verbatim text.
```
This    is    some    verbatim text.

Normally, using \\ will create a line
break, but not inside
a verbatim environment.
```

Note that `verbatim` text is always displayed using a monospaced font such as “courier”.

11.14 Mathematics mode

Since LaTeX was designed to typeset mathematical formulae, it is worth discussing how this is done. First, we need to put LaTeX into “math mode”. This is a state where many of the normal text commands do not work, instead, we can use mathematics commands.

11.14.1 Inline mathematics

If we want to use mathematics in the middle of a paragraph, then we use the dollar sign `$` to start and stop mathematics mode. For example, the text:

LaTeXSource Code

```
1 The formula $y = mx + c$ defines a straight line.
```

will appear as:

The formula $y = mx + c$ defines a straight line.

11.14.2 Display mathematics

If we want to separate the mathematical formulae from the text so that they appear on a line by themselves, we can use the `displaymath` environment. For example, the text:

LaTeXSource Code

```
1 The formula:
2 \begin{displaymath}
3 y = mx + c
4 \end{displaymath}
5 can be used to describe a straight line.
```

will be displayed as:

The formula:

$$y = mx + c$$

can be used to describe a straight line.

11.14.3 Equation environment

If we want to number our equations so that we can refer to them later, then we use the `equation` environment. For example, the text:

LaTeX Source Code

```
The formula for a straight line is given by:
\begin{equation}
y = mx + c
\end{equation}
The formula for a quadratic is given by:
\begin{equation}
y = ax^2 + bx + c
\end{equation}
```

will be displayed as:

The formula for a straight line is given by:

$$y = mx + c \tag{11.1}$$

The formula for a quadratic is given by:

$$y = ax^2 + bx + c \tag{11.2}$$

Notice how the formulae are numbered automatically according to the chapter in which they belong.

11.15 Mathematics

When we are typesetting mathematical formulae using LaTeX, we use the curly braces to group together parts of the equation that we want to apply the same command to. For example, the command ^ is used to make the following text superscript (e.g. a number raised to the power of another number), so if we wanted to represent 2^{x+y}, then we would use `$2^{x + y}$`.

11.15.1 Greek letters

To represent lowercase Greek letters we use the commands `\alpha`, `\beta`, `\gamma`,....

To represent uppercase Greek letters we use the commands `\Alpha`, `\Beta`, `\Gamma`,....

11.15.2 Exponents and subscripts

An exponent is specified with the caret character (^). A subscript is specified by the underscore character (_). For example, the text:

LaTeXSource Code

```
$a_i$                   \\
$x^2$                   \\
${a_i}^{i + j}$
```

will be displayed as:

a_i
x^2
${a_i}^{i+j}$

11.15.3 Square roots

A square root is added with the command `\sqrt`. An n^{th} root is created using `\sqrt[n]`. For example, the text:

LaTeXSource Code

```
$\sqrt{x^2 + y^2}$ and $\sqrt[3]{x}$
```

will be displayed as:

$\sqrt{x^2+y^2}$ and $\sqrt[3]{x}$

11.15.4 Fractions

A fraction is created using the command `\frac{`**numerator**`}{`**denominator**`}`. For example, the text:

LaTeXSource Code

```
$1\frac{1}{2}$ and $\frac{x^2}{x^2 + y^2}$
```

will be displayed as:

$$1\tfrac{1}{2} \text{ and } \frac{x^2}{x^2+y^2}$$

11.15.5 Other common operators

The integral operator is created with the command `\int`, and the sum operator is created with the command `\sum`. For example, the text:

LaTeXSource Code

```
\begin{displaymath}
\sum_{i=1}^n
\end{displaymath}
and
\begin{displaymath}
\int_{0}^{\frac{\pi}{2}}
\end{displaymath}
```

will be displayed as:

$$\sum_{i=1}^{n}$$

and

$$\int_{0}^{\frac{\pi}{2}}$$

11.16 Adding functionality with packages

LaTeX is designed to be easily extended. It is possible to define new commands and new environments reasonably easily. Although it is not difficult to do so, we will not discuss the creation of new commands and environments here.

However, we can add functionality by making use of the commands that other people have defined. These are distributed in files called "packages". There are thousands of different packages available on the Internet, but they need to be downloaded and installed before they can be used. Most installations of LaTeX already have most of the common packages installed so that they can be used immediately.

In order to use a package, we simply add the command:

`\usepackage{`**package name**`}`

at the top of the file (after the `\documentclass` declaration), in the preamble.

11.16.1 Graphicx package

The `graphicx` package allows us to import and use images in our documents. First, we need to add the command in the preamble that allows us to use the package as shown below:

LaTeXSource Code

```
\documentclass[a4paper]{article}
\usepackage{graphicx}

\begin{document}
...
```

Once we have specified to use the package, we can use the `includegraphics` command. This command is used to insert a picture at the location that the command appears. The format for this command is:

`\includegraphics[`**options**`]{`**file name**`}`

The options allow us to specify a height or width of the image. The image will be scaled so that it fits in the specified size. For example:

LaTeXSource Code

```
\includegraphics[height=10cm]{Photo}
```

Note that by default, the only file format for images that can be read by LaTeX is encapsulated postscript (). Most modern image processing software can save images in this format.

11.17 References

A number of resources have been used in the creation of this document.

- Essential LaTeX, Jon Warbrick, 2002
- Essential Mathematical LaTeX, D. Carlisle and R. Kaye, 1998
- The (Not So) Short Introduction to LaTeX 2_ε, Tobias Oetiker, 2003
- http://www.tug.org/interest.html
- http://www.latex-project.org/
- http://www.ctan.org

The tables showing the use of mathematical symbols at the end of this document have been reproduced from “Essential Mathematical LaTeX”, by Carlisle and Kaye.

α	\alpha	θ	\theta	o	o	τ	\tau
β	\beta	ϑ	\vartheta	π	\pi	υ	\upsilon
γ	\gamma	γ	\gamma	ϖ	\varpi	ϕ	\phi
δ	\delta	κ	\kappa	ρ	\rho	φ	\varphi
ε	\epsilon	λ	\lambda	ρ	\varrho	χ	\chi
ε	\varepsilon	μ	\mu	σ	\sigma	ψ	\psi
ζ	\zeta	ν	\nu	ς	\varsigma	ω	\omega
η	\eta	ξ	\xi				
Γ	\Gamma	Λ	\Lambda	Σ	\Sigma	Ψ	\Psi
Δ	\Delta	Ξ	\Xi	Υ	\Upsilon	Ω	\Omega
Θ	\Theta	Π	\Pi	Φ	\Phi		

Table 11.1: Greek Letters

$\pm$	\pm	$\cap$	\cap	$\diamond$	\diamond	$\oplus$	\oplus
$\mp$	\mp	$\cup$	\cup	$\triangle$	\bigtriangleup	$\ominus$	\ominus
$\times$	\times	$\uplus$	\uplus	$\triangledown$	\bigtriangledown	$\otimes$	\otimes
$\div$	\div	$\sqcap$	\sqcap	$\triangleleft$	\triangleleft	$\oslash$	\oslash
$*$	\ast	$\sqcup$	\sqcup	$\triangleright$	\triangleright	$\odot$	\odot
$\star$	\star	$\vee$	\vee	$\bigcirc$	\bigcirc	$\blacksquare$	\amalg
$\circ$	\circ	$\wedge$	\wedge	$\dagger$	\dagger	$\wr$	\wr
$\bullet$	\bullet	$\setminus$	\setminus	$\ddagger$	\ddagger	$\cdot$	\cdot
$+$	+	$-$	-				

Table 11.2: Binary Operation Symbols

$\leq$	\leq	$\geq$	\geq	$\equiv$	\equiv	$\models$	\models
$\prec$	\prec	$\succ$	\succ	$\sim$	\sim	$\perp$	\perp
$\preceq$	\preceq	$\succeq$	\succeq	$\simeq$	\simeq	$\mid$	\mid
$\ll$	\ll	$\gg$	\gg	$\asymp$	\asymp	$\parallel$	\parallel
$\subset$	\subset	$\supset$	\supset	$\approx$	\approx	$\bowtie$	\bowtie
$\subseteq$	\subseteq	$\supseteq$	\supseteq	$\cong$	\cong	$\propto$	\propto
$\neq$	\neq	$\smile$	\smile	$\vdash$	\vdash	$\dashv$	\dashv
$\sqsubseteq$	\sqsubseteq	$\sqsupseteq$	\sqsupseteq	$\doteq$	\doteq	$\frown$	\frown
$\in$	\in	$\ni$	\ni	$=$	=	$>$	>
$<$	<	:	:				

Table 11.3: Relation Symbols

,	,	;	;	:	\colon	.	\ldotp	$\cdot$	\cdotp

Table 11.4: Punctuation Symbols

$\leftarrow$	\leftarrow	$\longleftarrow$	\longleftarrow	$\uparrow$	\uparrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow	$\Uparrow$	\Uparrow
$\rightarrow$	\rightarrow	$\longrightarrow$	\longrightarrow	$\downarrow$	\downarrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow	$\Downarrow$	\Downarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow	$\updownarrow$	\updownarrow
$\Leftrightarrow$	\Leftrightarrow	$\Longleftrightarrow$	\Longleftrightarrow	$\Updownarrow$	\Updownarrow
$\mapsto$	\mapsto	$\longmapsto$	\longmapsto	$\nearrow$	\nearrow
$\hookleftarrow$	\hookleftarrow	$\hookrightarrow$	\hookrightarrow	$\searrow$	\searrow
$\leftharpoonup$	\leftharpoonup	$\rightharpoonup$	\rightharpoonup	$\swarrow$	\swarrow
$\leftharpoondown$	\leftharpoondown	$\rightharpoondown$	\rightharpoondown	$\nwarrow$	\nwarrow

Table 11.5: Arrow Symbols

$\ldots$	\ldots	$\cdots$	\cdots	$\vdots$	\vdots	$\ddots$	\ddots
$\aleph$	\aleph	$\prime$	\prime	$\forall$	\forall	∞	\infty
$\hbar$	\hbar	$\emptyset$	\emptyset	$\exists$	\exists	$\spadesuit$	\spadesuit
$\imath$	\imath	∇	\nabla	$\neg$	\neg	$\heartsuit$	\heartsuit
$\jmath$	\jmath	$\surd$	\surd	$\flat$	\flat	$\diamondsuit$	\diamondsuit
ℓ	\ell	$\top$	\top	$\natural$	\natural	$\clubsuit$	\clubsuit
$\wp$	\wp	$\bot$	\bot	$\sharp$	\sharp	∂	\partial
$\Re$	\Re	$\|$	|	\	\backslash	$\triangle$	\triangle
$\Im$	\Im	$\angle$	\angle	.	.	$\|$	|

Table 11.6: Miscellaneous Symbols

$\sum$	\sum	$\bigcap$	\bigcap	$\bigodot$	\bigodot
$\prod$	\prod	$\bigcup$	\bigcup	$\bigotimes$	\bigotimes
$\coprod$	\coprod	$\bigsqcup$	\bigsqcup	$\bigoplus$	\bigoplus
$\int$	\int	$\bigvee$	\bigvee	$\biguplus$	\biguplus
$\oint$	\oint	$\bigwedge$	\bigwedge		

Table 11.7: Variable-sized Symbols

\arccos	\cos	\csc	\exp	\ker	\limsup	\min	\sinh
\arcsin	\cosh	\deg	\gcd	\lg	\ln	\Pr	\sup
\arctan	\cot	\det	\hom	\lim	\log	\sec	\tan
\arg	\coth	\dim	\inf	\liminf	\max	\sin	\tanh

Table 11.8: Log-like Symbols

(	(	)	)	$\uparrow$	\uparrow	$\Uparrow$	\Uparrow
[	[	]	]	$\downarrow$	\downarrow	$\Downarrow$	\Downarrow
{	\{	}	\}	$\updownarrow$	\updownarrow	$\Updownarrow$	\Updownarrow
$\lfloor$	\lfloor	$\rfloor$	\rfloor	$\lceil$	\lceil	$\rceil$	\rceil
$\langle$	\langle	$\rangle$	\rangle	/	/	\	\backslash
$\|$	|	$\|\|$	\\|				

Table 11.9: Delimiters

⟅	\rmoustache	⟆	\lmoustache	⟯	\rgroup	⟮	\lgroup
⎪	\arrowvert	‖	\Arrowvert	⎪	\bracevert		

Table 11.10: Large Delimiters

$\hat{a}$	\hat{a}	$\acute{a}$	\acute{a}	$\bar{a}$	\bar{a}	$\dot{a}$	\dot{a}	$\breve{a}$	\breve{a}
$\check{a}$	\check{a}	$\grave{a}$	\grave{a}	$\vec{a}$	\vec{a}	$\ddot{a}$	\ddot{a}	$\tilde{a}$	\tilde{a}

Table 11.11: Math mode accents

$\widetilde{abc}$	\widetilde{abc}	$\widehat{abc}$	\widehat{abc}
$\overleftarrow{abc}$	\overleftarrow{abc}	$\overrightarrow{abc}$	\overrightarrow{abc}
$\overline{abc}$	\overline{abc}	$\underline{abc}$	\underline{abc}
$\overbrace{abc}$	\overbrace{abc}	$\underbrace{abc}$	\underbrace{abc}
$\sqrt{abc}$	\sqrt{abc}	$\sqrt[n]{abc}$	\sqrt[n]{abc}
f'	f'	$\frac{abc}{xyz}$	\frac{abc}{xyz}

Table 11.12: Some other constructions

CHAPTER 12

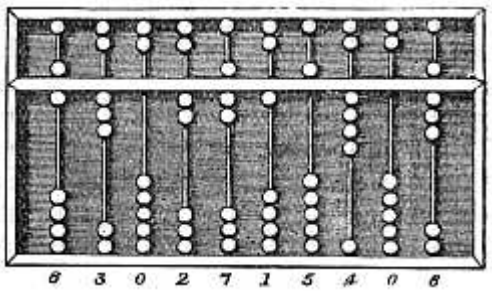
Development of the personal computer

12.1 Early history

The development of the computer has a long history, starting with the first machines that helped humans do arithmetic. Some of the most important contributions towards the development of electronic computers are listed here.

12.1.1 Abacus (1000–500BC)

The abacus was the first aid to calculation, possibly invented in Babylonia (now Iraq). The improved speed of calculation ensured the success of the abacus, which is still used today in some countries.



For more information, see:

- http://en.wikipedia.org/wiki/Abacus

12.1.2 Arabic numerals

Arabic numerals were introduced to Europe (800-1000 AD). The Arabic system included a number for zero, and greatly simplified calculations. It is the decimal system we still use today.

European	0	1	2	3	4	5	6	7	8	9
Arabic-Indic	٠	١	٢	٣	٤	٥	٦	٧	٨	٩
Eastern Arabic-Indic (Persian and Urdu)	٠	١	٢	٣	۴	۵	۶	٧	٨	٩
Devanagari (Hindi)	०	१	२	३	४	५	६	७	८	९
Tamil		க	உ	ங	ச	ஞ	ண	த	ந	ன

For more information, see:

- http://en.wikipedia.org/wiki/Arabic_Numerals

12.1.3 Wilhelm Schickard (1592–1635)

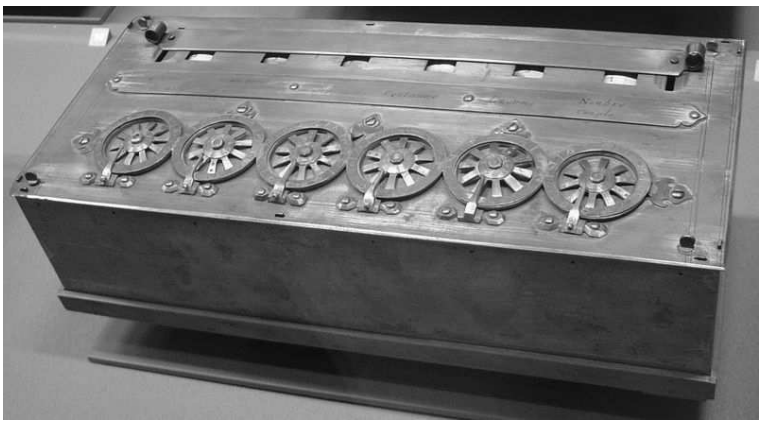
Wilhelm Schickard built the first known automatic calculator in 1623. It was known at the time as a mechanical calculating clock. It could add and subtract numbers up to six digits. It never made it past the prototype stage.

For more information, see:

- http://en.wikipedia.org/wiki/Wilhelm_Schickard

12.1.4 Blaise Pascal (1623–1662)

Blaise Pascal was the son of a tax collector. He spent many hours involved in mathematical operations, which inspired him to build a mechanical calculator in 1642. Over the next 10 years, Pascal built over 50 calculating machines and sold over a dozen. These had the capacity for eight digits, and could do both addition and subtraction. It was gear driven, and had a tendency to jam.

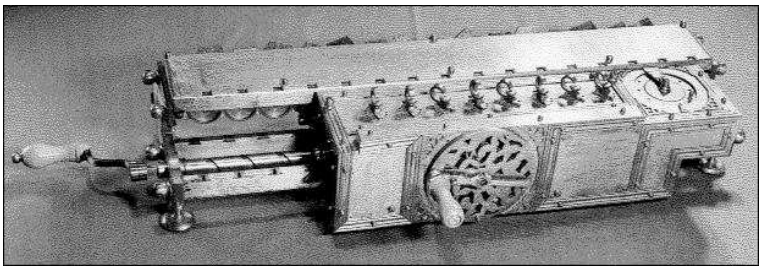


For more information, see:

- http://en.wikipedia.org/wiki/Pascal%27s_calculator

12.1.5 Gottfried Wilhelm von Leibniz (1646 - 1716)

Leibniz followed Pascal and built a digital calculating machine. It was gear and lever driven, and could do multiplication, division, addition, subtraction, and even calculate square roots using a series of additions. This device was called the “Stepped reckoner”. It was however, somewhat unreliable.

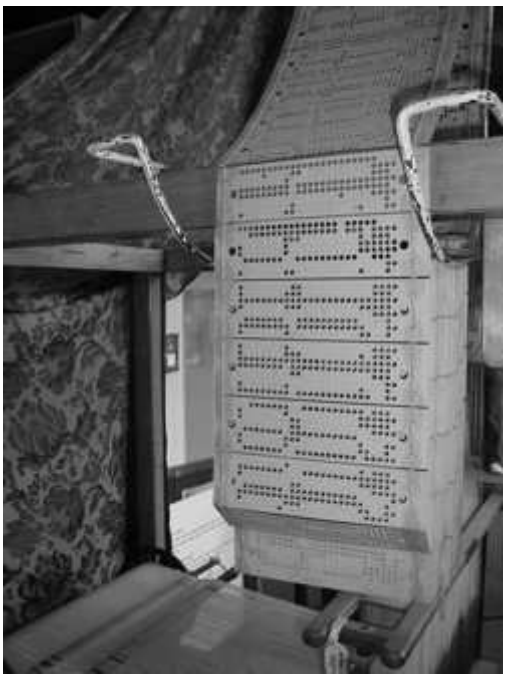


For more information, see:

- http://en.wikipedia.org/wiki/Stepped_Reckoner

12.1.6 Joseph Jacquard (1752 - 1834)

Jacquard built an automatic weaving loom, which used punch cards to control the selection of threads for weaving into complex patterns. Although it did not do any actual computation, it is considered to be an important step in the development of the computing machine because the punch cards controlled the operations that were done. This is an important concept that led to the design of the Babbage difference engine.

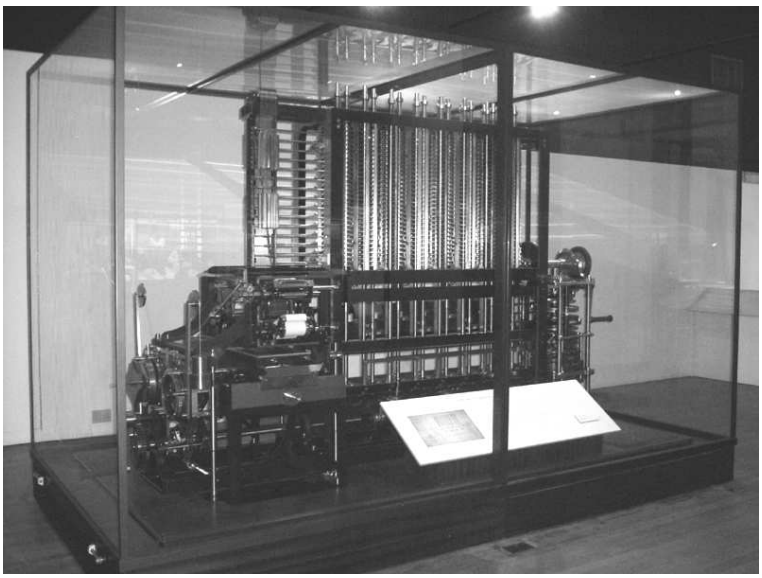


For more information, see:

- http://en.wikipedia.org/wiki/Jacquard_loom

12.1.7 Charles Babbage (1791 - 1871)

Charles Babbage, a mathematics professor of Cambridge, inspired by the Jacquard Loom, designed an automatic calculating machine called a “difference engine”. In 1822 he had a working model, which was to be fully automated, and steam powered. He lost interest in 1833 and began designing another more general machine called an Analytic Engine. It was never completed, partially due to a lack of engineering precision.



For more information, see:

- http://en.wikipedia.org/wiki/Charles_Babbage

12.1.8 Ada Augusta Lovelace (1816 - 1852)

Ada Augusta, Countess of Lovelace and daughter of Lord Byron is generally considered to be the first computer programmer. She corrected some of Babbage's errors and added her own ideas about the calculating machine. She is credited as developing the programming loop. However, there is some debate about the degree of her contribution as she occupies a rather sensitive place as the first significant female figure in the history of computing.

For more information, see:

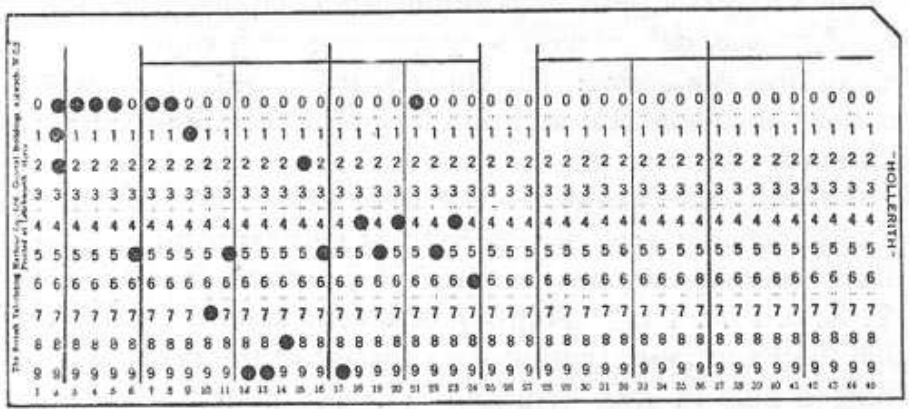
- http://en.wikipedia.org/wiki/Augusta_Ada_Lovelace

12.2 The electronic computer

The most important advances towards the electronic computer occurred during World War II. Although significant advances were made in America, Britain and Germany, only those made in the US continued to substantially influence computer development after the war.

12.2.1 Dr. Herman Hollerith (1860 - 1929)

Hollerith recognised a problem in the US Census Office where he worked. The 1880 US Census took 7 years to tabulate, and the population was steadily increasing. It was estimated that the 1890 Census would not be tabulated before starting on the 1900 census. He developed an electro-mechanical punch card tabulator, which automatically totalled the cards.



The 1890 Census was tabulated in 3 years, and Hollerith formed the Tabulating Machine Company. Thomas Watson joined the company in 1914, and with Watson as president, it was renamed International Business Machines (IBM) in 1924.

For more information, see:

- http://en.wikipedia.org/wiki/IBM

12.2.2 Atanasoff-Berry Computer (ABC)

John Vincent Atanasoff and Clifford E. Berry developed a special purpose computer that was designed to solve linear equations. The project to build the machine started in 1937 and it was successfully demonstrated before the project was officially cancelled in 1942. It was considered to be the first electronic computer, and it used the binary number system. It was, however, not programmable as later computers were.

For more information, see:

- http://en.wikipedia.org/wiki/Atanasoff-Berry_Computer

12.2.3 Z3

Konrad Zuse, a German engineer built a machine known as the Z3 in 1941. It was the first general purpose computer. It used binary mathematics and was fully capable of performing any computation possible. It was programmed using paper tape, but used mechanical relays, so it was not fully electronic. It was destroyed by an allied bombing raid on Berlin in 1944. Due to his nationality and association with the Nazi war machine, Zuse has not been given the recognition that he deserved as the creator of the first truly general purpose computing machine.

For more information, see:

- http://en.wikipedia.org/wiki/Konrad_Zuse

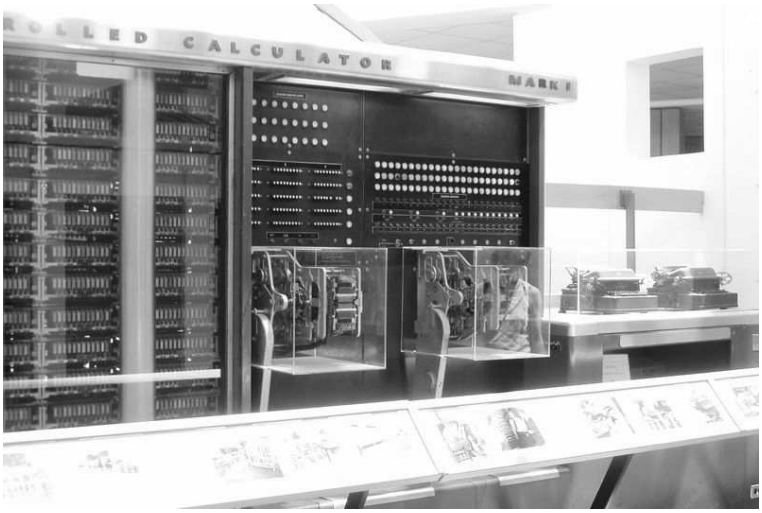
12.2.4 Colossus Mark I

The Colossus Mark I was designed by Tommy Flowers as part of the British war effort. The Colossus was an electronic computer designed to help break the German Enigma codes. It was operational at Bletchley Park in 1944.

After the war, Winston Churchill ordered the Colossus computers to be dismantled. The blueprints were burned and all information about the machines was classified as a state secret. It was not until the Official Secrets Act ended in 1976 that information about Bletchley Park and the Colossus machines became publicly available. Therefore, the design of the Colossus machines had little direct impact on the development of the modern computing machine.

12.2.5 Harvard Mark I

Prof. Howard Aiken sought backing from Watson in 1939, and with IBM support, the Harvard Mark I was produced in 1944. It was the first large scale digital computer produced in America. It used relays (electromagnetic switches) instead of gears, which was a large step forward. It took approximately 3 seconds to multiply two numbers together. Although it was programmable, it was limited by the lack of support for a conditional branch statement (i.e. it did not have the capacity to perform IF statements).

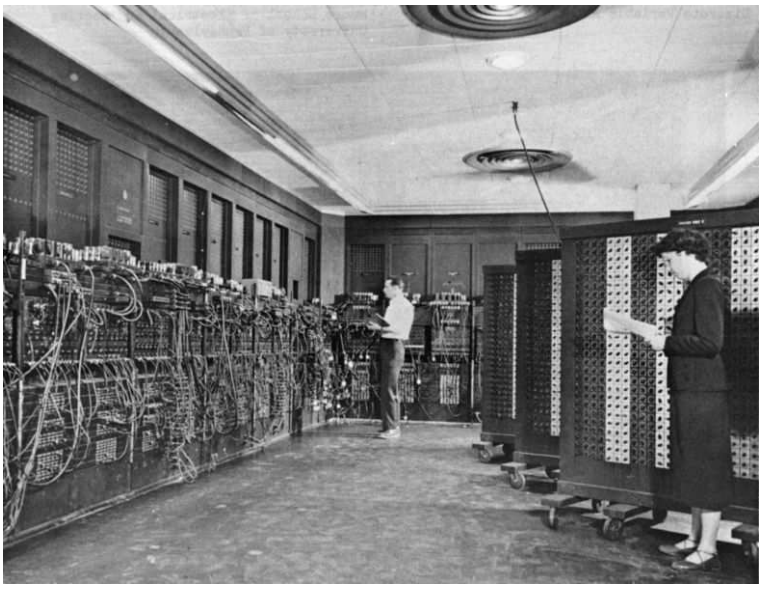


For more information, see:

- http://en.wikipedia.org/wiki/Harvard_Mark_I

12.2.6 ENIAC

John Mauchly viewed a demonstration of the ABC computer in 1941 and later began development (with John Eckert) of the ENIAC (Electronic Numerical Integrator and Computer), which was completed in 1946 under Army sponsorship. The ENIAC used vacuum tubes which were 1000 times faster than relays. It was 100 feet long, 10 feet high, and 3 feet wide, but could do a multiplication in 3 milliseconds. This was the first fully electronic computer that had a complete set of instructions, and could perform any computation. Some programs required the machine to be rewired. They later created the UNIVAC I.



For more information, see:

- http://en.wikipedia.org/wiki/ENIAC

12.2.7 John von Neumann (1903 - 1957)

John von Neumann who worked with both Eckert and Mauchly formed the Institute for Advanced Studies (IAS). He developed a general purpose computing machine, the IAS Computer (1945), which was capable of being programmed. Von Neumann is credited with the design of a computer system where the program was stored in main memory, the same place as the data. This design, known as the von Neumann architecture is the basis for the computers that we use today.

For more information, see:

- http://en.wikipedia.org/wiki/Von_Neumann

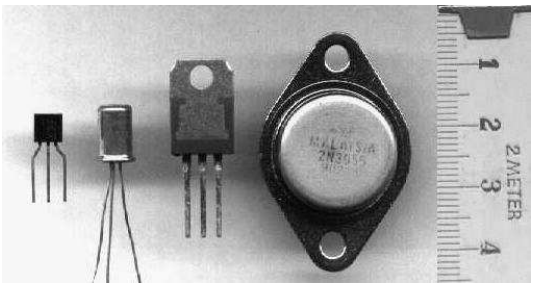
12.3 Commercialisation

In 1952, IBM released its first commercial electronic computer. Computers continued to play an important part in military and academic institutions, but over the next 20 years, computers began to be used in business. The telephone companies were one of the first industries to embrace computer technology, using computers to route communications signals.

As technology developed, the size of computer decreased. The first generation of computers used vacuum tubes (1951 - 1958) which were large and unreliable.

The invention of the transistor in 1947 by Bell Telephone Lab resulted in the old glass vacuum tubes (3 - 5cm long) being replaced with a small, cheap and reliable electronic

component (approx. 0.5 cm long), and defined the second generation of computers (1959 - 1964).



Later, the invention of the Integrated Circuit in 1959 by Texas Instruments/Fairchild Semiconductors started the third generation of computers (1965 - 1971). These silicon chips stored over 1000 transistors on a single piece of silicon. Finally, the fourth generation of computers began around 1971 with the use of large scale integrated circuits (LSI) and very large scale integrated circuits (VLSI). It was shortly afterwards that the personal computer industry began.

12.4 The personal computer industry

The personal computer industry is unique in the business world in many ways. It grew incredibly quickly from its beginning in 1975 to a billion dollar industry within 5 years. The industry was created and controlled by kids who had no formal business training. It is an industry where products must be recreated every 18 months, and competition is intense. It is an industry in which anyone can become a millionaire.

12.4.1 Mainframes

Computers were once large complex machines that were only affordable to large businesses. Large companies built the computers, sold them directly to the customer (business or government), serviced them for a monthly fee, and wrote the software which they licensed to the customer for another monthly fee. The computer maker made as much money from the post-sales servicing as they did from selling it in the first place.

There was only a small market for these machines, primarily in research institutions, government departments or very big businesses. An ordinary person had to get permission to even get close to them, and had to pay for the time spent using them. Many of these computers were controlled by IBM.

12.4.2 Xerox

In 1969, Xerox opened the Palo Alto Research Center (PARC). During the early 70's, Xerox decided not to enter the computing market. At this time they were already being investigated for monopolistic business practices. By 1977, half of Xerox's revenue was spent on defence in court. However, Xerox PARC's research contributed significantly

to many of the advances in computer technology, including Ethernet, WYSIWYG, the development of the GUI (as we know it), the invention of Laser Printers and the development of Smalltalk (the first functional language).

12.4.3 Intel

The first integrated circuit was announced by Fairchild Semiconductors in 1959. Nine years later, in 1968, Robert Noyce and Gorden Moore left Fairchild Semiconductors and formed Intel Corporation. Their microprocessors became more and more powerful, and in 1974 they created the Intel 8080 chip. This microprocessor had all the components needed for an entire computer. These chips were available to anyone, and the time was right for people to build their own machines.

12.5 The first personal computer—Altair 8800

Ed Roberts was interested in computers, but could not afford to own one. Like many other enthusiasts, he wanted to build his own computer with the new affordable microprocessors from Intel. He ran a small calculator company called MITS, but nobody was buying his calculators, and MITS was going bankrupt. Ed hoped to save MITS by marketing a kitset computer, and the bank reluctantly agreed to loan him the $65,000 he needed. He was considered an optimist for expecting to sell 800 in a year. The first Altair kitset appeared on the cover of Popular Electronics in January 1975. Within a month he was receiving 250 orders a day. The personal computer industry was born. Ed Roberts assembled 40,000 Altairs before he sold the business in 1978 when it became just too competitive.

12.5.1 Microsoft

Paul Allen and Bill Gates had been friends since high school, and had already had experience writing software for mainframe computers. The picture on the Popular Electronics cover excited Allen and Gates who realised that there would be a market for software, and a lot of money could be made. Fearing they were already too late, they wrote a version of BASIC which would work on the 8080 chip. Ed Roberts was shown a

demo after which he agreed to package the BASIC language with the Altair. Gates quit Harvard University and together with Allen formed Micro-Soft (later renamed Microsoft).

Paul Allen (left) and Bill Gates (right)

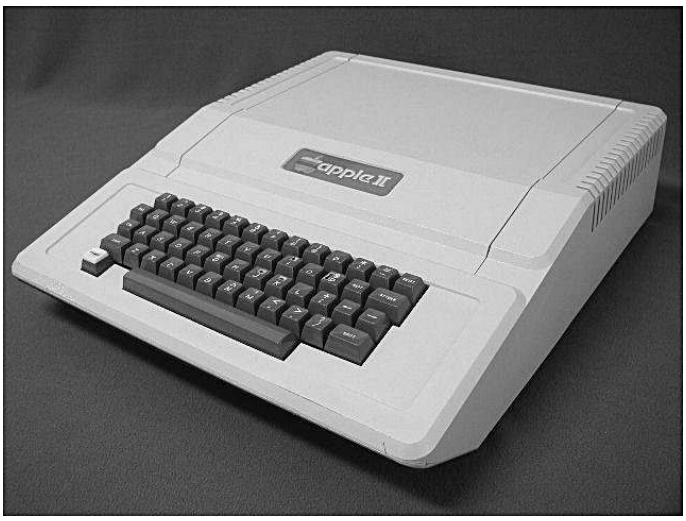
12.5.2 Homebrew Computer Club

The Altair took about 40 hours to put together, and even then it didn't always work. If it did work, you ended up with a box with a row of switches and a set of lights. There was no keyboard, no screen, and no storage device for information. It was difficult to use, so people formed clubs where they could discuss problems, and show off new developments. The Homebrew Computer Club was one such club in which everyone shared their solutions and helped each other to learn. It was here that Steve Wozniak and Steve Jobs first met.

12.6 Apple

Steve Wozniak was a hardware genius who began building his own computer at the Homebrew Computer Club. His technical ability attracted Steve Jobs who lacked the expertise of Wozniak, but had vision, drive and charisma.

Wozniak's first computer was called the Apple I, and consisted of a single circuit board without even a case. Steve Jobs managed to sell 50 Apple I's which convinced him that there was a market for a personal computer. His dream was to make computing available to everyone, at an affordable price. Some venture capital made the dream a possibility. Steve Wozniak designed the hardware, and Steve Jobs did the rest, demanding the Apple II looked like a piece of consumer electronics.



The Apple II was completed in 1977 and launched in 1978 at the West Coast Computer Faire. Two years later when Apple became a public company, both Steve Jobs and Steve Wozniak became millionaires.

12.6.1 VisiCalc

Apple computers initially had trouble finding a market. Businesses used mainframes, and nobody had a use for a computer at home, so sales of the Apple II were limited to enthusiasts, that is, until VISICALC.

Dan Bricklin and Bob Frankston invented what they called a visible calculator. It was a program which helped in financial planning. A table was created where the value in each cell in the table was related to the others. This meant that changing the value in one cell altered the value in the other cells accordingly. Today we call such programs spreadsheets.

After a slow start, businessmen everywhere became excited about the spreadsheets, and they all had to have it. VISICALC was only available on the Apple, and it was so useful that it justified buying an Apple computer. Sales soared.

12.7 IBM PC

By 1980, the personal computer market was worth over 1 billion dollars. IBM wanted to enter the market, but the company was designed to build very large scale computers, not small ones. The internal structuring of the company meant that it would take too long to develop a product, and it could not compete with smaller companies such as Apple.

Bill Lowe proposed a daring plan. To save time, they would not build a computer from scratch, but instead would buy existing components from other companies and assemble them. This concept of "open architecture" was alien to IBM, and difficult to sell to the corporate executives, but with the backing of the Chairman, Bill Lowe's team developed an IBM PC within a year. The IBM XT was released in 1983 and sold for $3,000.

IBM realised that competitors could copy the same architecture, but knew that they would always be able to buy in bulk, and thought that the bulk discount would ensure that they could always produce the machines for a lower price than competitors. The only remaining step to secure entry to the PC market was software development. A computer needs an operating system and a language used to write programs before it can be really useful, and IBM had neither.

12.7.1 CPM

Gary Kildall was a PhD graduate who programmed mainframes. During the early 70's he wrote an operating system for himself, and found that others were interested in purchasing it, so he formed a company called Digital Research. As personal computers were developed, Gary produced a version of his operating system known as CPM for them. By 1980, he had already sold 600,000 copies, so he was the obvious choice for IBM. However, he was arrogant, and refused to sign a contract with IBM because of a non-disclosure clause. IBM looked for alternatives.

12.7.2 Microsoft DOS

At the time IBM needed software, Microsoft was already the biggest supplier of compilers (programs that allow you translate from a high level programming language to a machine language that the CPU understands), but they had never written an operating system. IBM was willing to buy languages from Microsoft, but not without an operating system to run them on.

Fearing that they would lose the contract, Bill Gates promised to produce an operating system for IBM, but did not have time to write one himself. The solution was provided by Tim Patterson, a programmer who wrote an operating system based on CPM. He called it QDOS, and sold it to Seattle Computer Products. Bill Gates bought the licence for QDOS from SCP for $50,000. Two days later they handed it over to IBM under the name MS-DOS.

Microsoft was paid a fixed fee (about $80,000) with no royalties for both MS-DOS and BASIC. In itself, this deal wasn't worth much, but the key to Microsoft's success was that IBM had no control over the licensing of the software to other people. Microsoft expected other people to build machines compatible with the IBM PC to whom they could license their software. And that is exactly what happened.

12.7.3 Clones

The chips used in IBM's open architecture were made by Intel. These chips were sold to anyone who was interested, and many people were. In 1982, one year after the IBM PC was shipped, a group of engineers got together and formed a company called Compaq in order to create a computer compatible with the IBM PC. They bought the same chips from Intel, and by reverse engineering, produced a computer which behaved the same as an IBM PC, but was a little cheaper.

In the first year of business, Compaq sales reached $111 million. Soon, there were many companies repeating the process and producing their own clones, always a little cheaper than IBM. Driving the sales of both the IBM PC and all the clones was another spreadsheet. Based on VISICALC, Lotus 1-2-3 provided a spreadsheet for the IBM PC. Within a year, Lotus was worth $150 million, and you no longer needed to buy an Apple II.

12.7.4 Compaq 386

Once IBM had entered the personal computer industry, it threatened to dominate the entire market. The early success of Apple computers was beginning to fade in the mid 80's and IBM held 50% of the market. People were concerned that IBM would swallow the personal computer market and maintain a complete monopoly over computer technology.

The turning point for IBM came in September 1986 when Compaq released a new IBM compatible computer based on the new Intel 80386 chip. This computer was released before IBM had released its own version based on the same chip. Compaq showed the world that IBM could be beaten, and others could compete against "Big Blue".

The prices of the clones kept falling, and IBM could not keep pace. By the early 90's, IBM was losing enormous amounts of money (5-6 million dollars a day), and it retreated from the PC industry, defeated by its own open architecture design.

12.8 Apple Macintosh

In 1968 Doug Engelbart of the Stanford Research Institute publicly demonstrated a word processor which used windows to display the text. Xerox PARC developed this idea further by creating a graphical user interface for the computers they were using for research. These computers featured windows, pulldown menus, a mouse and a corresponding pointer for operating the system.

In 1979 Steve Jobs was given a tour of PARC, and it was this graphical interface which caught his attention. He had the vision to see that an easy to use interface (like the one in PARC) would open up computing to the masses, and allow everyone to share the computing experience.

Steve Jobs set about creating a computer that anyone could use. It began as the Lisa, and was later redesigned and renamed the Macintosh. It was released in January 1984. Over time, driven by Lotus 1-2-3 and backed by its good name, IBM began to overtake sales of the Apple II. The Macintosh had to be good, and it needed software. It needed something that IBM PC's couldn't do, and a man named John Warnock provided the solution.

12.8.1 Adobe

John Warnock had developed a new technology, which allowed laser printers to print exactly what was displayed on the screen. He left Xerox PARC and founded Adobe systems to develop the concept of WYSIWYG (what you see is what you get). Steve Jobs recognised the value of his work, and Apple invested in 20 percent of Adobe.

The quality of laser printed images, combined with Macintosh's ease of use created a brand new industry, desk-top publishing. The Macintosh had found its place in the

market. Apple was still in trouble however, but Steve Jobs would not admit it. He disagreed with the management of Apple, and in 1985 sold all his shares in Apple and left in disgust.

12.9 Microsoft Windows

Impressed by the GUI concept of the Macintosh, Microsoft began to build its own version, called Windows. The first couple of versions weren't very good, but then in 1990, Windows 3.0 was launched, and it was good enough to compete with the Macintosh.

People could now do the same things on a Macintosh that they did on an IBM compatible, but for a lower cost, since the IBM clones were much cheaper than the Macintosh. Apple computers became a fading influence in the PC industry. In August 1995 Microsoft released Windows 95 with a GUI that was extremely similar to the Macintosh, and Apple was consigned to a niche in the market.

Although Windows 95 was unstable and enormously difficult to install, it looked good and made the IBM Compatible PC easy to use for the first time.

12.10 Conclusion

The computer industry is driven by technology that gets more powerful, yet cheaper every day. In order for a computer to survive this terrible competition, it must have both hardware and software. The software available for a machine has an overriding influence upon the consumer. The IBM compatible computer has become so popular because of a self-reinforcing cycle in the marketplace. People write software for the most popular machine because they can sell more programs. The more software that is available, the more popular the machine will become. Good products are ignored or overshadowed by marketing, and market forces. Steve Jobs had a vision of every home owning a computer. Bill Gates had a vision that they would all run Microsoft software. Both visions have come true.

12.11 References

- Accidental Empires. Robert Cringely
- A Short History of the Computer, Jeremy Meyers